



ISTQB®



**Certified Tester
Foundation Level
Agile Tester**

December 2024
ISTQB Certified Tester Foundation Level Agile Tester

ISTQB Certified Tester Foundation Level Agile Tester

TEMA 1: Desarrollo Ágil de Software	3
1.1. Fundamentos de Desarrollo Ágil de Software.....	3
1.1.1. El Desarrollo Ágil de Software y el Manifiesto Ágil	3
1.1.2. Enfoque de Equipo Completo	4
1.1.3. Retroalimentación Temprana y Frecuente	4
1.2. Características de los Enfoques Ágiles.....	5
1.2.1. Enfoques de desarrollo ágil de software	5
1.2.2. Creación colaborativa de Historias de Usuario.....	8
1.2.3. Retrospectivas.....	8
1.2.4. Integración continua	9
1.2.5. Planificación de entregas e iteraciones	10
PREGUNTAS TEMA 1	11
TEMA 2: Principios, Prácticas y Procesos Fundamentales de Prueba Ágil	14
2.1. Diferencias entre probar en enfoques tradicionales y ágiles	14
2.1.1. Actividades de prueba y desarrollo	14
2.1.2. Productos de trabajo de un proyecto.....	14
2.1.3. Niveles de prueba.....	15
2.1.4. Prueba y Gestión de la Configuración	15
2.1.5. Opciones de organización para pruebas independientes.....	16
2.2 Situación de la prueba en proyectos ágiles	16
2.2.1. Comunicar el estado de la prueba	16
2.2.2. Gestión del riesgo de regresión	17
2.3 Rol y competencias de un probador ágil.....	17
2.3.1. Competencias de un probador ágil.....	17
2.3.2. Rol de un probador ágil	18
PREGUNTAS TEMA 2	19
TEMA 3: Métodos, Técnicas y Herramientas de Prueba Ágiles	22
3.1. Métodos de Prueba Ágil.....	22
3.1.1. Desarrollo Guiado por Pruebas, Desarrollo Guiado por Pruebas de Aceptación y Desarrollo Guiado por el Comportamiento.....	22

3.1.2. La Pirámide de Prueba	23
3.1.3. Cuadrantes de Prueba, Niveles de Prueba y Tipos de Prueba	24
3.1.4. El Rol de un Probador.....	24
3.2. Evaluación de los Riesgos de Calidad y Estimación del Esfuerzo de Prueba	26
3.2.1. Evaluación de los Riesgos de Calidad en Proyectos Ágiles.....	26
3.2.2. Estimación del Esfuerzo de Prueba en Función del Contenido y el Riesgo.....	27
3.3. Técnicas de Proyectos Ágiles	27
3.3.1. Criterios de Aceptación, Cobertura Adecuada y Otra Información para Probar.....	27
3.3.2. Aplicación del Desarrollo Guiado por Pruebas de Aceptación	31
3.3.3. Diseño de Pruebas de Caja Negra Funcionales y No Funcionales	32
3.3.4. Prueba Exploratoria y Prueba Ágil	32
3.4. Herramientas en Proyectos Ágiles	33
3.4.1. Herramientas de Gestión y Seguimiento de Tareas	33
3.4.2. Herramientas de Comunicación e Intercambio de Información.....	33
3.4.3. Herramientas de Construcción y Distribución de Software	34
3.4.4. Herramientas de Gestión de la Configuración	34
3.4.5. Herramientas de Diseño de Pruebas, Implementación y Ejecución	35
3.4.6. Informática en la Nube y Herramientas de Virtualización.....	35
PREGUNTAS TEMA 3	36

TEMA 1: Desarrollo Ágil de Software

1.1. Fundamentos de Desarrollo Ágil de Software

1.1.1. El Desarrollo Ágil de Software y el Manifiesto Ágil

El manifiesto ágil contiene **4 enunciados de valores**:

- **Individuos e interacciones sobre procesos y herramientas**
 - Desarrollo muy centrado en las personas
 - Trabajo a través de la comunicación e interacción continuas
- **Software funcionando sobre documentación extensiva**
 - Software en funcionamiento es más útil y valioso que documentación excesiva
 - Útil en entornos de negocio que cambian rápidamente y cuando se intenta innovar en un nuevo dominio de problemas
 - Proporciona más tiempo de comercialización
- **Colaboración con el cliente sobre negociación contractual**
 - Colaborar directamente con el cliente mejora la posibilidad de comprender exactamente lo que éste requiere
- **Respuesta ante el cambio sobre seguir un plan**
 - El cambio es inevitable en los proyectos de software
 - Es importante tener flexibilidad en las prácticas de trabajo para aceptar el cambio

El manifiesto ágil contiene **12 principios**:

La prioridad es satisfacer al cliente	Software como medida de proceso
Aceptar los cambios	Desarrollo sostenible
Entregar software frecuentemente	Excelencia técnica
Negocio y desarrolladores juntos	Simplicidad
Individuos motivados	Equipos autoorganizados
Conversación cara a cara	Reflexión para ser más efectivos

1.1.2. Enfoque de Equipo Completo

- Participación de todos los miembros
- Comunicación mediante reuniones de pie diarias
- Dinámica de equipo más eficaz y eficiente
- La calidad es responsabilidad de todo el equipo

Beneficios		
Mejorar la comunicación y la colaboración dentro del equipo	Permitir que se aprovechen las distintas competencias del equipo en beneficio del proyecto	Hacer que la calidad sea responsabilidad de todos

1.1.3. Retroalimentación Temprana y Frecuente

- Los proyectos ágiles tienen iteraciones cortas que permiten al equipo del proyecto recibir una retroalimentación temprana y continua sobre la calidad del producto a lo largo del ciclo de vida de desarrollo.
- Gracias a la retroalimentación frecuente, los equipos ágiles pueden incorporar la mayoría de los nuevos cambios en el proceso de desarrollo del producto.

Beneficios		
Evitar malentendidos sobre los requisitos	Aclarar las solicitudes de prestaciones de los clientes	Descubrir, aislar y resolver los problemas de calidad de forma temprana
Aportar información al equipo ágil sobre su productividad y capacidad para realizar entregas	Promover un impulso consistente al proyecto	

1.2. Características de los Enfoques Ágiles

1.2.1. Enfoques de desarrollo ágil de software

I. Programación extrema (XP)

Basado en valores, principios y prácticas.

Valores

- Comunicación
- Simplicidad
- Retroalimentación
- Valor
- Respeto

Principios	
Humanidad	Flujo
Economía	Oportunidad
Beneficio mutuo	Redundancia
Autosimilitud	Fallo
Mejora	Calidad
Diversidad	Pasos de bebé
Reflexión	Responsabilidad aceptada

Prácticas	
Sentarse juntos	Ciclo trimestral
Equipo completo	Holgura
Espacio de trabajo informativo	Construcción en 10 minutos
Trabajo energizado	Integración continua
Programación en pareja	Programar probando primero
Historias	Diseño incremental
Ciclo semana	

II. Scrum

Contiene varios instrumentos y prácticas.

- **Sprint:** Iteraciones de duración fija (entre 2 y 4 semanas).
- **Incremento de producto:** Producto obtenido tras una iteración que puede entregarse al cliente.
- **Trabajo acumulado del Producto:** Lista de elementos del producto, gestionada por el propietario del producto.
- **Trabajo acumulado del Sprint:** Lista de elementos que se realizarán en el sprint, gestionada por el equipo de desarrollo.
- **Definición de hecho:** Criterios para asegurar un producto potencialmente entregable.
- **Acotamiento del tiempo:** Seleccionar solamente las tareas que se puedan finalizar durante el sprint.
- **Transparencia:** El equipo informa sobre su progreso en las reuniones diarias.

Define varios roles

Scrum Máster	Product Owner	Equipo de desarrollo
Se asegura que las prácticas y reglas de Scrum se cumplan	Representa al cliente y prioriza la lista de trabajo acumulado	Se encarga de desarrollar y probar el producto. Las decisiones las toma el equipo al completo

III. Kanban

Su objetivo es visualizar y optimizar el flujo de trabajo dentro de una cadena de valor añadido. Se utilizan 3 instrumentos:

Tablero Kanban:

- Tablero con columnas que representan estados de las tareas.
- Cada tarea se representa como un ticket.
- Los tickets se van moviendo de izquierda a derecha en el tablero.

Limitación del trabajo en curso:

- Hay un número máximo de tareas activas en paralelo.

Plazo de ejecución:

- Se optimiza el flujo de tareas minimizando el plazo de ejecución.

SCRUM VS KANBAN	Scrum	Kanban
Visualizar tareas en un tablero	SI	SI
Lista de trabajo acumulado	SI	SI
Iteraciones o Sprints	SI	OPCIONAL
Posibilidad de entregar elemento por elemento	NO	SI
Acotamiento de tiempo	SI	OPCIONAL

1.2.2. Creación colaborativa de Historias de Usuario

- Las historias de usuario se escriben para **capturar los requisitos** desde la perspectiva de todo el equipo.
- Deben abordar tanto las características **funcionales** como las **no funcionales**.
- Cada historia incluye criterios de aceptación para estas características.
- Un equipo ágil considera que una tarea está terminada cuando se han **cumplido los criterios de aceptación**.

Una historia de usuario es la conjunción de **3 elementos**:

- **Tarjeta:** Soporte físico que describe una historia de usuario. La descripción debe ser precisa.
- **Conversación:** La conversación explica cómo se utilizará el software. La conversación puede estar documentada o ser verbal.
- **Confirmación:** Los criterios de aceptación, se utilizan para confirmar que la historia está hecha.

I	Independiente	Debe poder desarrollarse sin depender de otras historias
N	Negociable	Debe permitir ajustar antes de empezar a desarrollarse
V	Valioso	Debe aportar valor al negocio
E	Estimable	Debe ser estimable y realizable en un sprint.
S	Small	Debe ser pequeño y realizable dentro de un sprint.
T	Testeable	Debe poder probarse y confirmar si cumple los requisitos

1.2.3. Retrospectivas

- Reunión que se celebra al final de cada iteración para discutir sobre:
 - ¿Qué ha ido bien?
 - ¿Qué se puede mejorar?
 - ¿Cómo incorporamos las mejoras?
- Pueden dar lugar a decisiones de mejora relacionadas con la prueba, centradas en la efectividad de la prueba la productividad de la prueba, la calidad de los casos de prueba y la satisfacción del equipo.
- El momento y la organización de la retrospectiva dependen del método ágil concreto que se siga.

- Los probadores deberían desempeñar un papel importante en las retrospectivas. Los probadores forman parte del equipo y aportan su perspectiva única.

Todo el equipo puede contribuir a cualquier actividad, tanto si es de prueba como si no.

1.2.4. Integración continua

- Consiste en fusionar todos los cambios realizados en el software e integrar todos los componentes modificados regularmente, al menos una vez al día.
- Un proceso de integración continua consta de las siguientes actividades automatizadas:
 - **Análisis estático de código:** Llevar a cabo el análisis estático de código e informar los resultados.
 - **Compilación:** Compilar y enlazar el código, generando los archivos ejecutables.
 - **Prueba unitaria:** Ejecutar las pruebas unitarias, comprobar la cobertura de código e informar los resultados de la prueba.
 - **Despliegue:** Instalar el producto construido en un entorno de prueba.
 - **Prueba de integración:** Ejecutar las pruebas de integración e informar los resultados.
 - **Informe:** Publicar el estado de todas estas actividades en un lugar visible para el público.

Ventajas	
Detección y análisis en problemas de integración	Hace visible el progreso a todo el equipo
Aporta retroalimentación rápida sobre el código	Elimina riesgos de calendario asociados a la integración big bang
1 día de diferencia entre el software que se prueba y el que se desarrolla	Aporta disponibilidad constante de software durante todo el sprint
Reduce el riesgo de regresión tras la refactorización del código	Reduce las pruebas manuales
Aporta confianza sobre el desarrollo	Aporta retroalimentación rápida sobre las decisiones tomadas

Riesgos
Hay que introducir y mantener las herramientas
Hay que definir y establecer el proceso
Se requieren recursos adicionales
Se requiere una cobertura de prueba profunda
Exceso de confianza en las pruebas unitarias

1.2.5. Planificación de entregas e iteraciones

Planificación de entrega: se produce unos meses antes del inicio del proyecto. En esta etapa se define la **lista de trabajo acumulado**.

Los probadores participan en:

- **Definir historias de usuario** que puedan ser probadas, incluyendo los criterios de aceptación.
- **Análisis de riesgo** de proyecto y de calidad.
- Estimar el **esfuerzo de prueba** asociado a las historias de usuario.
- Definir los **niveles de prueba** necesarios.
- **Planificar la prueba** para la entrega.

Planificación de iteración: el equipo de desarrollo selecciona historias de usuario de la lista de trabajo acumulado y estima el esfuerzo que requiere cada una.

Los probadores participan en:

- Análisis de riesgo detallado de las historias de usuario.
- Determinar la capacidad de una historia de ser probada.
- Crear **pruebas de aceptación** para las historias de usuario.
- Descomponer las historias de usuario en **tareas**.
- Estimar el **esfuerzo de prueba** para todas las tareas de prueba.
- Identificar aspectos **funcionales y no funcionales** del sistema.
- Apoyar la **automatización** de pruebas.

Actividades comunes en ambas planificaciones:

- Definir el **alcance de la prueba**.
- Seleccionar los **miembros del equipo** que llevarán a cabo las actividades de prueba.
- Definir el **entorno de prueba y los datos de prueba** necesarios.
- Definir el **momento, la secuencia, las dependencias y los prerequisitos** para las actividades de prueba.
- Los **riesgos** de proyecto y de calidad que deben abordarse.

PREGUNTAS TEMA 1

1. Una User Story que fue considerada muy simple, resultó ser más compleja de lo que se estimó, requiriendo más esfuerzo para la implementación y el testing. ¿Qué planificación se verá afectada por esto?

- Planificación de la iteración**
- Planificación de la entrega
- Planificación de la iteración y entrega
- Ninguna planificación se verá afectada

Si una historia de usuario requiere más esfuerzo, significa que ese esfuerzo se obtiene de otras tareas de la misma iteración.

Un cambio en el esfuerzo requerido para una historia podría llegar a perjudicar a la planificación de la entrega si provocase retrasos reiteradamente. Pero, en principio, solo tiene impacto inmediato en la planificación de la iteración.

2. Según Scrum, ¿qué significa la palabra “lista de trabajo acumulado del producto” o “pila del producto”?

- La evolución de una lista de priorización de productos de ítems gestionada por el Propietario del producto
- Iteraciones divididas de 2 a 4 semanas
- Lista de priorización de ítems gestionada por el propietario de producto
- Tareas, requisitos y características que el equipo espera terminar dentro de un sprint**

La lista de trabajo acumulado contiene todos los elementos planificados para realizarse en vista a todo el proyecto, y es gestionado por el propietario del producto.

No confundir con la lista de trabajo acumulado del sprint, que contiene una serie de elementos seleccionados de la lista de trabajo acumulado que se van a realizar durante el sprint.

3. ¿Cómo optimiza *Kanban* el flujo continuo de tareas?

- Minimizando el tiempo de espera para el flujo de valor completo**
- Maximizando el tiempo de espera para el flujo de valor completo
- Haciendo constante el tiempo de espera para el flujo de valor completo para todas las iteraciones
- Kanban no optimiza el flujo continuo de tareas

En el enfoque Kanban, se le da especial importancia al flujo de las tareas. Se intenta que no haya interrupciones para así terminar el trabajo más rápido.

Eso significa que se intenta reducir ese “tiempo de espera”.

4. La retrospectiva se puede utilizar para evaluar la productividad de los evaluadores y, por lo tanto, realizar una auditoría para que el equipo sea más eficiente.

- Verdadero
- Falso**
- Depende del tipo de tester en el equipo
- Las retrospectivas pueden ser usadas para asistir la productividad de los desarrolladores, no de los testers

Las retrospectivas no se usan para evaluar el rendimiento de los miembros del equipo, ni para señalar culpables de los problemas.

La utilidad de las retrospectivas se basa en remarcar las cosas que han ido bien, y las que se pueden mejorar a futuro.

5. ¿Cómo ayuda la integración continua a proporcionar más confianza a los miembros del equipo y aportarles mayor visibilidad del progreso?

- Gracias a herramientas que automatizan tareas
- Gracias a que el código se analiza, compila, se despliega y se prueba frecuentemente con un sistema estable**
- Gracias a testers con gran conocimiento técnico
- Ninguna de las anteriores

La integración continua aporta ventajas al equipo gracias a todas las etapas que la componen.

Las demás opciones mencionadas son positivas para el proyecto, pero no están estrechamente relacionadas con la integración continua.

TEMA 2: Principios, Prácticas y Procesos Fundamentales de Prueba Ágil

2.1. Diferencias entre probar en enfoques tradicionales y ágiles

2.1.1. Actividades de prueba y desarrollo

- Las pruebas se realizan **durante toda la iteración**, no solamente al final
- **Todo el equipo** se involucra en las pruebas
- **Práctica de “Corregir errores primero”** -> solucionar los defectos de la anterior iteración al principio de la siguiente
- En la Programación Extrema (XP) -> **Programación en pareja**
- Los probadores actúan como **entrenadores de la prueba**
- Las **pruebas automatizadas** juegan un papel fundamental
- Las pruebas manuales se reducen a pruebas basadas en la experiencia del probador

¿Es recomendable hacer la práctica de “Corregir errores primero”?

No siempre. A veces puede provocar que se desconozca el trabajo total a realizar en la iteración, complicando así la estimación de las tareas.

2.1.2. Productos de trabajo de un proyecto

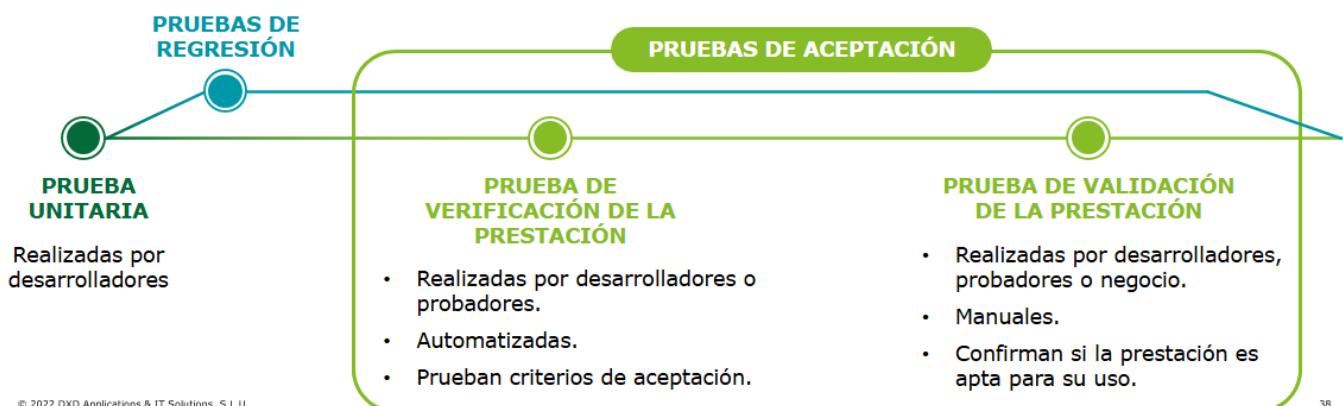
Los productos de trabajo del proyecto de interés directo para los probadores ágiles suelen pertenecer a tres categorías:

- De negocio: describen lo que se necesita y cómo utilizarlo
 - Especificaciones de requisitos
 - Documentación de usuario
- De desarrollo: describen cómo se construye el sistema
 - Diagramas de entidad-relación
 - Código
 - Pruebas unitarias
- De prueba: describen cómo se prueba el sistema
 - Planes de prueba
 - Pruebas automatizadas

ÉPICAS: colecciones de características relacionadas. Muchas historias de usuario sobre una misma funcionalidad conforman una épica.

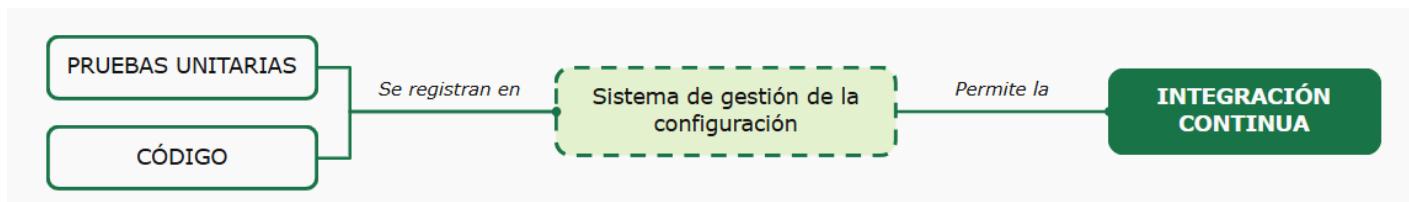
2.1.3. Niveles de prueba

- Los niveles de prueba son actividades de prueba que están relacionadas de forma lógica, a menudo por la madurez o completitud del elemento sujeto a prueba
- Modelos secuenciales: los niveles de prueba **NO se solapan**
- Modelos iterativos: si **PUEDEN solaparse**



2.1.4. Prueba y Gestión de la Configuración

- Uso intensivo de **herramientas automatizadas** para desarrollar, probar y gestionar el producto.



- Las pruebas automatizadas también pueden incluir **pruebas funcionales**, aunque se suelen separar de las unitarias por su duración.
- No se debe confiar demasiado en las pruebas unitarias, ya que no son tan efectivas detectando defecto. Se deben ejecutar también pruebas automatizadas de **integración y sistema**.
- El uso de la gestión de la configuración **resuelve el ciclo** de “Construcción – Instalación – Reconstrucción – Reinstalación”.

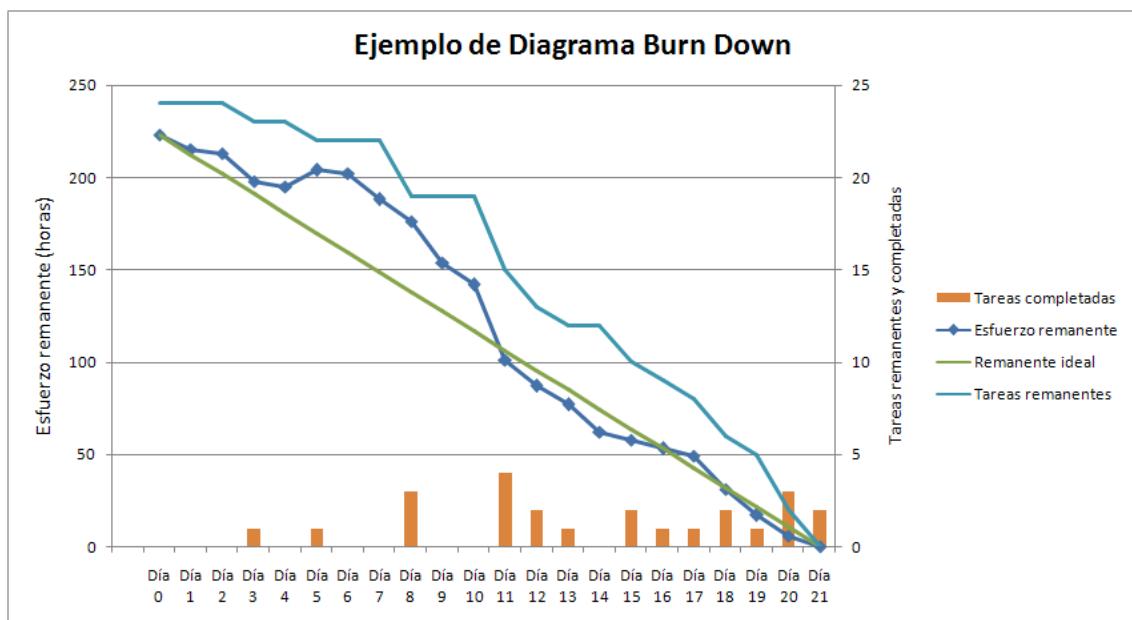
2.1.5. Opciones de organización para pruebas independientes

	Probadores integrados en el equipo y asignados al principio	Probadores separados y asignados al final	Probadores separados y asignados al principio
Independencia		X	X
Evaluación objetiva		X	X
Conocimiento del producto	X		X

2.2 Situación de la prueba en proyectos ágiles

2.2.1. Comunicar el estado de la prueba

- Recomendable usar **herramientas que generan informes automáticamente.**
- **Gráfico de quedado o burn-down:** representa la cantidad de trabajo restante (eje Y) respecto al tiempo (eje X)



Temas que tratar en las reuniones diarias:

- ¿Qué has completado desde la última reunión?
- ¿Qué tienes previsto completar para la próxima reunión?
- ¿Qué se está interponiendo en tu camino?

2.2.2. Gestión del riesgo de regresión

- El **riesgo de regresión** aumenta en cada iteración
- Recomendaciones para reducir el riesgo de regresión:
 - Automatizar todo lo que se pueda lo antes posible
 - Usar **herramientas de gestión de la configuración**
- **Pruebas de verificación de la construcción:** aquellas que cubren la funcionalidad y prueban puntos críticos del sistema.

Además de la automatización de la prueba, también se pueden automatizar las siguientes tareas de prueba:

- Generación de datos de prueba
- Carga de los datos de prueba en los sistemas
- Despliegue de construcciones en los entornos de prueba
- Restauración de un entorno de una línea de base
- Comparación de las salidas de datos

2.3 Rol y competencias de un probador ágil

2.3.1. Competencias de un probador ágil

- Ser positivos y estar orientados a las soluciones con los miembros del equipo y los implicados
- Mostrar un pensamiento crítico, orientado a la calidad y escéptico sobre el producto
- Recabar información de los implicados de forma activa
- Evaluar y comunicar con precisión los resultados de las pruebas, el avance de estas y la calidad del producto
- Trabajar eficazmente para definir historias de usuario que puedan ser probadas
- Colaborar dentro del equipo, trabajando en pareja con los programadores y otros miembros del equipo
- Responder rápidamente a los cambios, incluyendo la modificación, adición o mejora de los casos de prueba
- Planificar y organizar su propio trabajo

2.3.2. Rol de un probador ágil

- Comprender, implementar y actualizar la estrategia de prueba
- Medir e informar de la cobertura de la prueba en todas las dimensiones de cobertura aplicables
- Asegurar el uso adecuado de las herramientas de prueba
- Configurar, utilizar y gestionar los entornos de prueba y los datos de prueba
- Informar de los defectos y trabajar con el equipo para resolverlos
- Participar de forma proactiva en las retrospectivas del equipo
- Colaborar activamente con los desarrolladores y los implicados del negocio para aclarar los requisitos
- Asegurar que se programen las tareas de prueba adecuadas durante la planificación de la entrega y la iteración
- Entrenar a otros miembros del equipo en los aspectos relevantes de la prueba

PREGUNTAS TEMA 2

1. ¿Cuál es el inconveniente de la práctica de “Corregir errores primero”?

- Algunos errores pueden pasar desapercibidos
- No hay ningún inconveniente
- La longitud de la iteración se extenderá
- Da lugar a una situación en la que se desconoce la cantidad de trabajo total a realizar en la iteración**

Empezar una iteración corrigiendo los errores de la anterior puede provocar que no se sepa con certeza cuánto trabajo se debe realizar en la iteración.

2. En un proyecto ágil se están llevando a cabo pruebas de aceptación con las siguientes características:

- Las llevan a cabo desarrolladores y testers
- Son automatizadas
- Prueban los criterios de aceptación

¿A qué nivel de prueba se está haciendo referencia?

- Pruebas de verificación de la prestación**
- Pruebas de validación de la prestación
- Pruebas unitarias

Las pruebas de validación de la prestación pueden llevarse a cabo desde negocio, suelen ser manuales y confirman si la prestación es apta para su uso.

3. ¿Cuál de las siguientes afirmaciones es FALSA sobre el ciclo “Construcción – Instalación – Fallo – Reconstrucción – Reinstalación”?

- Es un ciclo ineficiente resuelto mediante la gestión de la configuración
- Es un ciclo ineficiente resuelto mediante el uso de Scrum**
- Los informes de prueba en tiempo real ayudan a resolverlo
- Es un ciclo ineficiente que ocurre en muchos proyectos tradicionales

El ciclo “Construcción – Instalación – Fallo – Reconstrucción – Reinstalación” es un ciclo ineficiente resuelto gracias a los informes de prueba en tiempo real que proporcionan las herramientas de gestión de la configuración.

No se resuelve usando la metodología Scrum.

4. ¿Cuál es el RIESGO de tener probadores integrados dentro del equipo asignados desde el principio del proyecto?

- Pérdida de independencia**
- Tienen menos conocimiento del producto
- Perjudica a la relación con los desarrolladores
- Sienten más presión de tiempo

Los probadores asignados desde el principio pierden independencia.

Al estar asignados desde el principio, sí tendrán un mayor conocimiento del producto, por lo que descartamos la B.

Al estar asignados desde el principio, tendrán una mejor relación con el resto del equipo, por lo que descartamos la C.

La presión de tiempo suele afectar a los probadores asignados al final del sprint, ya que no tienen conocimiento del producto, por lo que descartamos la D.

5. ¿Cuál de las siguientes opciones puede reducir la efectividad de la prueba?

- Dependencia excesiva de las pruebas unitarias
- Involucrar a probadores independientes en el proyecto
- Usar tableros de tareas y gráficos de evolución
- Ejecutar pruebas de aceptación diariamente

Uno de los riesgos principales del uso de automatización de pruebas es confiar demasiado en las pruebas unitarias.

Las pruebas unitarias no son infalibles para detectar defectos. Es necesario realizar también pruebas de integración y de sistema.

TEMA 3: Métodos, Técnicas y Herramientas de Prueba Ágiles

3.1. Métodos de Prueba Ágil

Tipos de desarrollo: El desarrollo guiado por pruebas, el desarrollo guiado por pruebas de aceptación y el desarrollo guiado por el comportamiento son tres técnicas complementarias que se utilizan en equipos ágiles para llevar a cabo las pruebas en los distintos niveles de prueba.

Tipos		
Desarrollo guiado por pruebas	Desarrollo guiado por pruebas de aceptación	Desarrollo guiado por el comportamiento BDD
TDD	ATDD	BDD

3.1.1. Desarrollo Guiado por Pruebas, Desarrollo Guiado por Pruebas de Aceptación y Desarrollo Guiado por el Comportamiento

Desarrollo guiado por pruebas TDD: se utiliza para desarrollar código guiado por casos de prueba automatizados.

- Se añade una prueba que capture el concepto del programador sobre el funcionamiento deseado de un pequeño fragmento de código.
- Se ejecuta la prueba, que debería fallar ya que el código no existe.
- Se escribe el código y se ejecuta la prueba en un bucle cerrado hasta que la prueba pase
- Se refactoriza el código después de que la prueba haya sido superada, y vuelve a ejecutar la prueba para asegurarse de que sigue pasando contra el código refactorizado
- Se repite este proceso para el siguiente pequeño fragmento de código, ejecutando las pruebas anteriores, así como las pruebas añadidas

Desarrollo guiado por pruebas de aceptación ATDD

- Define los criterios de aceptación y las pruebas durante la creación de historias de usuario
- Es un enfoque colaborativo que permite a todos los implicados entender cómo tiene que comportarse el componente de software y qué necesitan los desarrolladores, probadores y representantes de negocio para garantizar ese comportamiento
- Crea pruebas reutilizables para las pruebas de regresión

- Permite resolver rápidamente los defectos y validar el comportamiento de las prestaciones

Desarrollo guiado por el comportamiento BDD

- Permite al desarrollador concentrarse en probar el código basándose en el comportamiento esperado del software desde la perspectiva del usuario y no en las funcionalidades de este
- Las pruebas suelen ser más fáciles de entender para los demás miembros del equipo y los implicados
- Marcos de desarrollo guiados por comportamiento basados en formato

Dado un contexto inicial,	Given some initial context,
Cuando se produce un evento,	When an event occurs,
Entonces se aseguran algunos resultados.	Then ensure some outcomes

3.1.2. La Pirámide de Prueba

Un sistema de software puede ser probado en diferentes niveles. Los niveles de prueba típicos son, desde la base de la pirámide hasta la cima:

- Unidad
- Integración
- Sistema
- Aceptación



La pirámide de prueba hace hincapié en tener un **gran número de pruebas en los niveles inferiores** (base de la pirámide) y, a medida que el desarrollo avanza hacia los niveles superiores, el número de pruebas disminuye (cima de la pirámide).

El concepto de pirámide de prueba se basa en el principio de control de calidad y pruebas tempranas.

3.1.3. Cuadrantes de Prueba, Niveles de Prueba y Tipos de Prueba

Los cuadrantes de prueba alinean los niveles de prueba con los tipos de prueba adecuados en la metodología ágil.

Las pruebas pueden ser totalmente manuales, totalmente automatizadas o una combinación de manuales y automatizadas.

Q1	Q2
<ul style="list-style-type: none"> • Es el nivel unitario • Apoya a los desarrolladores • Contiene pruebas unitarias • Estas pruebas deben automatizarse 	<ul style="list-style-type: none"> • Es el nivel sistema • De cara al negocio, comportamiento • Contiene pruebas funcionales • Comprueban los criterios de aceptación • Pueden ser manuales o automatizadas
Q3	Q4
<ul style="list-style-type: none"> • Es el nivel de aceptación de sistema • De cara al negocio • Contiene pruebas exploratorias • Estas pruebas suelen ser manuales y están orientadas al usuario 	<ul style="list-style-type: none"> • Es el nivel de aceptación de operativa • Orientado a la tecnología • Contiene pruebas de rendimiento, carga, estrés y escabilidad, seguridad, mantenibilidad, memoria, etc. • Estas pruebas suelen estar automatizadas

3.1.4. El Rol de un Probador

Trabajo en Equipo: buenas prácticas.

- Interdisciplinario
- Autoorganizado
- Ubicación común
- Colaborativo
- Empoderado
- Comprometido
- Transparente
- Creíble
- Abierto a la retroalimentación
- Resiliente

Esprint cero: es la primera iteración del proyecto en la que tienen lugar muchas actividades de preparación. El probador colabora con el equipo en las siguientes actividades durante esta iteración.

- Identificar el alcance del proyecto
- Crear una arquitectura inicial del sistema y prototipos de alto nivel
- Planificar, adquirir e instalar las herramientas necesarias
- Realice un análisis del riesgo de calidad inicial
- Definir las métricas de prueba para medir el proceso de prueba
- Especificar la definición de hecho
- Cree el tablero de tareas
- Definir cuándo continuar o detener la prueba antes de entregar el sistema al cliente

Integración

- En los proyectos ágiles, el objetivo es entregar valor al cliente de forma continua
- La estrategia de integración debe considerar tanto el diseño como la prueba
- Es importante identificar todas las dependencias entre las funciones y características subyacentes

Planificación de la prueba

- Dado que las pruebas están totalmente integradas en el equipo ágil, la planificación de la prueba debe comenzar durante la sesión de planificación de la entrega y actualizarse durante cada esprint
- La planificación del esprint da como resultado un conjunto de tareas que se colocan en el tablero de tareas, donde cada tarea debe tener una duración de uno o dos días de trabajo

Prácticas de Prueba Ágil

- Trabajo en pareja: dos miembros del equipo se sientan juntos en un puesto de trabajo para realizar una prueba y otra tarea del sprint
- Diseño de prueba incremental: los casos de prueba y los contratos se construyen gradualmente a partir de las historias de usuario y otras bases de prueba

- Mapa Mental: los mapas mentales son una herramienta útil a la hora de probar

3.2. Evaluación de los Riesgos de Calidad y Estimación del Esfuerzo de Prueba

3.2.1. Evaluación de los Riesgos de Calidad en Proyectos Ágiles

Uno de los muchos retos de la prueba es la selección, asignación y priorización adecuadas de las condiciones de prueba.

En los proyectos ágiles, el análisis del riesgo de calidad tiene lugar en dos puntos:

- **Planificación de la entrega:** los representantes de negocio que conocen las prestaciones de la entrega proporcionan una visión general de alto nivel de los riesgos, y todo el equipo, incluidos los probadores, pueden ayudar en la identificación y evaluación del riesgo
- **Planificación de la iteración:** todo el equipo identifica y evalúa los riesgos de calidad

Las tareas se priorizan en función del nivel de riesgo de calidad asociado a ellas.

Tareas con riesgos más altos -> Empiezan antes y mayor esfuerzo

Tareas con riesgos más bajos -> Empiezan más tarde y menor esfuerzo

En los siguientes pasos se expone un ejemplo de cómo puede llevarse a cabo el proceso de análisis del riesgo de calidad en un proyecto ágil durante **la planificación de la iteración**:

- Reunir a los miembros del equipo ágil, incluido el probador o probadores
- Hacer una lista de todos los elementos de la lista de trabajo acumulado para la iteración actual (por ejemplo, en un tablero de tareas)
- Identificar los riesgos de calidad asociados a cada elemento, teniendo en cuenta todas las características de calidad pertinentes
- Evaluar cada riesgo identificado, lo que incluye dos actividades: categorizar el riesgo y determinar su nivel de riesgo basándose en el impacto y la probabilidad de defectos
- Determinar el alcance de la prueba de forma proporcional al nivel de riesgo
- Seleccionar las técnicas de prueba adecuadas para mitigar cada riesgo, basándose en el riesgo, el nivel de riesgo y la característica de calidad pertinente

3.2.2. Estimación del Esfuerzo de Prueba en Función del Contenido y el Riesgo

- Durante la planificación de la entrega, el equipo ágil estima el esfuerzo necesario para completar la entrega. La estimación aborda también el esfuerzo de prueba. Una técnica de estimación habitual en los proyectos ágiles es el póker de planificación, una técnica basada en el consenso.
- Los valores representan el número de puntos de historia, días de esfuerzo u otras unidades en las que el equipo estima.
- Una estimación elevada suele significar que la historia no se entiende bien o que debe dividirse en varias historias más pequeñas.
- Los estimadores discuten la prestación y hacen preguntas al propietario de producto si es necesario.
- Es aconsejable incluir el nivel de riesgo de un elemento de la lista de trabajo acumulado, además de la prioridad especificada por el propietario del producto

3.3. Técnicas de Proyectos Ágiles

3.3.1. Criterios de Aceptación, Cobertura Adecuada y Otra Información para Probar

Los proyectos ágiles esbozan los **requisitos iniciales** como **historias de usuario** en una lista de trabajo acumulado priorizada al comienzo del proyecto. Los requisitos iniciales son **breves** y suelen seguir un **formato predefinido**.

Las **historias de usuario** sirven como una importante **base de prueba**. Otras posibles bases de prueba son:

- Experiencia de proyectos anteriores
- Funciones, prestaciones y características de calidad existentes en el sistema
- Código, arquitectura y diseño
- Perfiles de usuario
- Información sobre defectos de proyectos existentes y anteriores
- Una categorización de los defectos en una taxonomía de defectos
- Estándares aplicables
- Riesgos de calidad

Para que se pueda probar, los criterios de aceptación deben abordar los siguientes temas cuando sean relevantes:

- **Comportamiento funcional:** el comportamiento observable externamente con las acciones del usuario como entrada que opera bajo determinadas configuraciones
- **Características de calidad:** la forma en que el sistema realiza el comportamiento especificado. Las características también pueden denominarse atributos de calidad o requisitos no funcionales.
- **Escenarios (casos de uso):** una secuencia de acciones entre un actor externo (a menudo un usuario) y el sistema, con el fin de lograr un objetivo específico o una tarea de negocio.
- **Reglas de negocio:** actividades que sólo pueden realizarse en el sistema bajo ciertas condiciones definidas por procedimientos y restricciones externas
- **Interfaces externas:** descripciones de las conexiones entre el sistema que se van a desarrollar y el mundo exterior. Las interfaces externas pueden dividirse en diferentes tipos
- **Restricciones:** cualquier restricción de diseño e implementación que limite las opciones del desarrollador
- **Definiciones de datos:** el cliente puede describir el formato, el tipo de datos, los valores permitidos y los valores por defecto de un elemento de datos en la composición de una estructura de datos de negocio compleja

Definición de **hecho** de los **Niveles de Prueba**:



- **Pruebas de Sistema:**

- Pruebas de extremo a extremo de las historias de usuario
- Todas las personas usuarias cubiertas
- Cubiertas las características de calidad más importantes del sistema
- Todos los riesgos de calidad cubiertos según el alcance acordado de la prueba
- Todas las pruebas de regresión automatizadas
- Todos los defectos encontrados se informan y posiblemente se solucionan
- No hay defectos importantes sin resolver

- **Pruebas de Integración:**

- Todos los requisitos funcionales probados
- Todas las interfaces entre unidades probadas
- Todos los riesgos de calidad cubiertos según el alcance acordado de la prueba
- Ningún defecto importante sin resolver
- Todos los defectos detectados se comunican
- Todas las pruebas de regresión automatizadas, cuando sea posible, con todas las pruebas automatizadas almacenadas en un repositorio común

- **Prueba Unitaria:**

- 100% de cobertura de decisión siempre que sea posible
- Análisis estático realizado en todo el código
- Ningún defecto importante sin resolver
- Ninguna deuda técnica inaceptable conocida en el diseño y el código
- Todo el código, pruebas unitarias y resultados de pruebas unitarias revisados
- Todas las pruebas unitarias automatizadas
- Las características importantes están dentro de los límites acordados

Definición de **hecho** para las **Historias de Usuario**:

- Las historias de usuario seleccionadas para la iteración están completas, son entendidas por el equipo y tienen criterios de aceptación detallados y que pueden ser probados
- Se han especificado y revisado todos los elementos de la historia de usuario, incluidas las pruebas de aceptación de la historia de usuario
- Las tareas necesarias para implementar y probar las historias de usuario seleccionadas han sido identificadas y estimadas por el equipo

Definición de **hecho** para las **Prestaciones**:

- Todas las historias de usuario que la componen, con criterios de aceptación, están definidas y aprobadas por el cliente
- El diseño está completo, sin deuda técnica conocida
- El código está completo, sin deuda técnica conocida ni refactorización inacabada
- Se ha realizado la prueba unitaria y se ha alcanzado el nivel de cobertura definido
- Se ha realizado la prueba de integración y prueba de sistema para la prestación de acuerdo con los criterios de cobertura definidos
- No queda defectos por corregir
- La documentación de la prestación está completa, lo que puede incluir notas de la publicación, manuales de usuario y funciones de ayuda en línea

Definición de **hecho** para las **Iteraciones**:

- Todas las prestaciones de la iteración están listas y se han probado individualmente según los criterios del nivel de la prestación
- Cualquier defecto no crítico que no pueda solucionarse dentro de las limitaciones de la iteración se añade a la lista de trabajo acumulado del producto y se prioriza
- Integración de todas las prestaciones para la iteración completada y probada
- Documentación redactada, revisada y aprobada
- En este punto, el software es potencialmente liberable porque la iteración se ha completado con éxito, pero no todas las iteraciones dan como resultado una entrega

Definición de **hecho** para las **Entregas**:

- **Cobertura:** todos los elementos relevantes de la base de prueba para todos los contenidos de la entrega han sido cubiertos por la prueba
- **Calidad:** la intensidad de los defectos, la densidad de defectos, el número estimado de defectos restantes están dentro de los límites aceptables, las consecuencias de los defectos no resueltos y restantes se entienden y son aceptables, el nivel de riesgo residual asociado a cada riesgo de calidad identificado se entiende y es aceptable
- **Tiempo:** si se ha alcanzado la fecha de entrega predeterminada, hay que tener en cuenta las consideraciones de negocio asociadas a la entrega y a la no entrega
- **Coste:** el coste estimado del ciclo de vida debe utilizarse para calcular el retorno de la inversión del sistema entregado

3.3.2. Aplicación del Desarrollo Guiado por Pruebas de Aceptación

- El desarrollo guiado por pruebas de aceptación es un enfoque que da prioridad a las pruebas
- Los casos de prueba se crean antes de implementar la historia de usuario
- Los casos de prueba son creados por el equipo ágil, que incluye al desarrollador, al probador y a los representantes de negocio y pueden ser manuales o automatizados
- El siguiente paso es crear las pruebas. Esto puede hacerlo el equipo en conjunto o el probador individualmente. En cualquier caso, una persona independiente, como un representante de negocio, valida las pruebas
- Las pruebas son ejemplos que describen las características específicas de la historia de usuario

3.3.3. Diseño de Pruebas de Caja Negra Funcionales y No Funcionales

- En la prueba Ágil, los probadores crean muchas pruebas de forma simultánea a las actividades de programación de los desarrolladores
- Al igual que los desarrolladores programan basándose en las historias de usuario y los criterios de aceptación, los probadores crean pruebas basadas en las historias de usuario y sus criterios de aceptación.
- Técnicas de diseño de caja negra:
 - Análisis de particiones de equivalencia
 - Análisis del valor frontera
 - Tablas de decisión
 - Pruebas de transición de estado.
- En muchas situaciones, los requisitos no funcionales pueden documentarse como historias de usuario

3.3.4. Prueba Exploratoria y Prueba Ágil

Prueba exploratoria:

- La prueba exploratoria es importante en los proyectos ágiles debido al escaso tiempo disponible para el análisis de prueba y los detalles limitados de las historias de usuario
- El diseño y la ejecución ocurren al mismo tiempo, guiados por un contrato de prueba establecido. Un contrato de prueba proporciona las condiciones de prueba que se deben cubrir durante una sesión de prueba con límite de tiempo



- La gestión de prueba exploratoria basada en sesiones, entre 60 y 120 minutos
- Capacidad de plantearse preguntas sobre lo que hay que probar
- El probador utiliza la creatividad, intuición, conocimiento y competencia del software, dominio del negocio
- Conjunto de heurísticas (Fronteras, CRUD, Interrupciones)
- Documentar el proceso lo máximo posible
- Registrar la información en herramientas de gestión para facilitar la comprensión del estado de todas las pruebas a los implicados

3.4. Herramientas en Proyectos Ágiles

3.4.1. Herramientas de Gestión y Seguimiento de Tareas

Competencias

- Registrar las historias y sus correspondientes tareas de desarrollo y prueba
- Capturar las estimaciones de los miembros del equipo sobre sus tareas y calcular automáticamente el esfuerzo necesario para implementar una historia
- Asociar las tareas de desarrollo y las tareas de prueba con la misma historia
- Agregar las actualizaciones de los desarrolladores y probadores al estado de la tarea a medida que completan su trabajo
- Proporcionar una representación visual del estado actual de cada historia de usuario, la iteración y la entrega, lo que permite a todos los implicados
- Integración con herramientas de gestión de la configuración, que pueden permitir el registro de entrada de código

3.4.2. Herramientas de Comunicación e Intercambio de Información

Wikis

- Diagramas de las prestaciones del producto, discusiones sobre las prestaciones, diagramas de los prototipos, fotos de las discusiones en la pizarra, y otra información
- Herramientas y/o técnicas de desarrollo y prueba que otros miembros del equipo consideran útiles
- Métricas, gráficos y paneles de control sobre el estado del producto, lo que resulta especialmente útil cuando el wiki está integrado con otras herramientas, como el servidor de construcción y el sistema de gestión de

tareas, ya que la herramienta puede actualizar el estado del producto automáticamente

- Conversaciones entre los miembros del equipo, similares a la mensajería instantánea y al correo electrónico, pero de forma compartida con todos los miembros del equipo

Mensajería instantánea

- Las herramientas de mensajería instantánea, teleconferencia de audio y videochat ofrecen las siguientes ventajas
- Involucrar a los equipos distribuidos en las reuniones de pie
- Reducir las facturas telefónicas mediante el uso de la tecnología de voz sobre IP, eliminando las limitaciones de costes que podrían reducir la comunicación de los miembros del equipo en entornos distribuidos

Escritorio compartido

- En equipos distribuidos, se pueden realizar demostraciones de productos, revisiones de código e incluso trabajo en pareja
- Captura de demostraciones de productos al final de cada iteración, que pueden publicarse en la wiki del equipo. Estas herramientas deben utilizarse para complementar y ampliar, no para sustituir, la comunicación cara a cara en los equipos ágiles

3.4.3. Herramientas de Construcción y Distribución de Software

La construcción diaria y el despliegue de software es una práctica clave en los equipos ágiles. Esto requiere el uso de herramientas de integración continua y de distribución de la construcción.

3.4.4. Herramientas de Gestión de la Configuración

En los equipos ágiles, las herramientas de gestión de la configuración pueden utilizarse no sólo para almacenar el código fuente y las pruebas automatizadas, sino que las pruebas manuales y otros productos de trabajo de prueba suelen almacenarse en el mismo repositorio que el código fuente del producto.

El tamaño del equipo, la estructura, la ubicación y los requisitos de integración con otras herramientas determinarán qué sistema de control de versiones es el adecuado para un proyecto ágil concreto.

3.4.5. Herramientas de Diseño de Pruebas, Implementación y Ejecución

Aunque la mayoría de estas herramientas no son nuevas o específicas para proyectos ágiles, proporcionan importantes capacidades dado el cambio que se produce en los proyectos ágiles:

Herramientas de diseño de pruebas:

- El uso de herramientas como los mapas mentales se ha hecho muy popular para diseñar y definir rápidamente las pruebas de una nueva prestación

Herramientas de gestión de casos de prueba:

- El tipo de herramientas de gestión de casos de prueba utilizadas en un entorno ágil pueden formar parte de la gestión del ciclo de vida de la aplicación

Herramientas de preparación y generación de datos de prueba:

- Las herramientas que generan datos para poblar la base de datos de una aplicación son muy beneficiosas cuando se necesitan muchos datos

Herramientas de carga de datos de prueba:

- Una vez generados los datos para las pruebas, hay que cargarlos en la aplicación. La introducción manual de datos suele requerir mucho tiempo y es propensa a errores

Herramientas de ejecución de pruebas automatizadas:

- Existen herramientas de ejecución de prueba que están más alineadas con las pruebas ágiles.

Herramientas de prueba exploratoria:

- Las herramientas que capturan y registran las actividades realizadas en una aplicación durante una sesión de prueba exploratoria

3.4.6. Informática en la Nube y Herramientas de Virtualización

- La virtualización permite que un único recurso físico (servidor) funcione como muchos recursos separados y más pequeños
- Cuando se utilizan máquinas virtuales o instancias en la nube, los equipos disponen de un mayor número de servidores para el desarrollo y las pruebas
- Esto puede ayudar a evitar los retrasos asociados a la espera de servidores físicos

PREGUNTAS TEMA 3

1. En la sesión de planificación de la versión, el evaluador comunica las ideas de la prueba utilizando cuadrantes de prueba para que el equipo comprenda el propósito de todos los tipos y niveles de prueba incluidos en el ciclo de vida del desarrollo. De las siguientes, ¿qué afirmaciones del evaluador sobre los cuadrantes de prueba son verdaderas?

- El cuadrante Q1 son pruebas de nivel de unidad y el cuadrante Q4 son pruebas de aceptación de usuario
- El cuadrante Q1 son pruebas de nivel de unidad y el cuadrante Q4 son pruebas funcionales
- El cuadrante Q3 son pruebas de rendimiento y el cuadrante Q4 son pruebas funcionales
- El cuadrante Q2 son pruebas funcionales y el cuadrante Q4 son pruebas de rendimiento**

No hay explicación, búscate la vida. 😊

2. Para mantener la visibilidad, ¿qué puede configurar un probador durante el sprint cero?

- i. Crear tablero de tareas.
- ii. Crear gráficos de evolución.
- iii. Especificar la definición de hecho.
- iv. Definir cuándo continuar o detener las pruebas antes de la entrega.

- i, ii, iii
- ii, iii, iv
- i, ii, iv
- i, iii, iv

No hay explicación, búscate la vida. 😊

3. ¿Qué técnica permite a un desarrollador centrarse en probar el código en función de comportamiento esperado del software?

- TDD
- BDD**
- ATDD
- XP

No hay explicación, búscate la vida. 😊

4. Un equipo ágil ha realizado todas las tareas necesarias para identificar riesgos de calidad, categorizar riesgos y evaluar el nivel de riesgo, estimar los esfuerzos de acuerdo con el nivel de riesgo y la estrategia seleccionada para mitigar los riesgos. En medio de la iteración, el cliente presentó un conjunto de nuevos requisitos y cambios en las historias de usuario seleccionadas para la iteración actual. ¿Qué debe hacer el equipo en términos de mantener el riesgo bajo?

- Dado que los cambios se produjeron en medio de la iteración, posponga el cambio a la siguiente iteración
- Volver a evaluar el riesgo identificando el nuevo riesgo, el nivel de riesgo y la estrategia de mitigación**
- Si los nuevos requisitos son elementos de alto riesgo, solo entonces se pueden incluir en la iteración actual
- Detener las tareas seleccionadas para la iteración y comenzar a trabajar en los nuevos cambios para finalizar al final de la iteración

No hay explicación, búscate la vida. 😊

5. ¿Cuáles de los siguientes son ejemplos de criterios de aceptación comprobables?

- i. La interfaz de usuario debe verse bien
- ii. La aplicación web debe ser funcional en los principales navegadores: Firefox, Chrome, Safari e IE
- iii. Un usuario no puede comprar hasta que su dirección de correo electrónico haya sido verificada.
- iv. La mayoría de los probadores deben realizar sus propias pruebas exploratorias en el sistema.
- v. La operación de inicio de sesión nunca debe exceder los 0,5 segundos.

- i, ii, v
- i, iii, v
- ii, iii, iv
- ii, iii, v

No hay explicación, búscate la vida. 😊

6. ¿Qué herramienta será útil para proporcionar una instantánea del servidor para que los desarrolladores resuelvan errores?

- Herramienta de virtualización y computación en la nube
- Herramienta de captura de video
- Panel de control de estilo wiki
- Herramientas de gestión de casos de prueba

No hay explicación, búscate la vida. 😊