Desarrollo Web Entorno Desarrollo

Fase de Pruebas

¿Qué es Probar?

Probar es una forma de evaluar la calidad del producto y reducir el riesgo de fallos en un entorno de operaciones o en producción.

El proceso de prueba aparte de realizar las pruebas, también incluye actividades tales como planificar la prueba, analizar, diseñar e implementar pruebas, informar del avance y de los resultados de la prueba, y evaluar la calidad de un objeto de prueba.

Objetivos Característicos de la Prueba

Para cualquier proyecto dado, los objetivos de prueba pueden incluir:

- Evaluar productos de trabajo tales como requisitos, historias de usuario, diseño y código.
- Verificar el cumplimiento de todos los requisitos especificados.
- Validar si el objeto de prueba está completo y funciona como los usuarios y otros implicados esperan.
- Generar confianza en el nivel de calidad del objeto de prueba.
- Prevenir defectos.
- Encontrar fallos y defectos.
- Proporcionar suficiente información a los implicados para que puedan tomar decisiones informadas, especialmente en relación con el nivel de calidad del objeto de prueba.
- Reducir el nivel de riesgo de calidad inadecuada del producto.
- Cumplir con requisitos o normas contractuales, legales o reglamentarias, y/o verificar el cumplimiento de dichos requisitos o normas por parte del objeto de prueba.

Conceptos de pruebas

- **Error**: Desviación entre el comportamiento real y el comportamiento esperado, no cumplimiento de un requisito establecido.
- **Defecto**: Anomalía en un componente o sistema que puede dar lugar a que no se lleve a cabo correctamente una función determinada. El término “bug” se aplica históricamente a los defectos en informática.
- **Fallo**: Manifestación de un defecto cuando el diseño de la prueba no se encuentra correctamente definido.
- **Equivocación**: Acción humana que da lugar a un resultado incorrecto.
- **Enmascaramiento del error**: Varios estados de error se compensan mutuamente, es decir, no aparece reflejado el error porque otros los contrarrestan.
- **Depuración**: Localización y corrección de errores internos.
- **Prueba**: Búsqueda dirigida y sistemática de los efectos del error, para demostrar defectos.
- **Caso de prueba**: Unión de una prueba y unas condiciones de entorno establecidas, por ejemplo, requisitos de ejecución, datos de entrada fijos en relación con los datos esperados o con un comportamiento o miles de casos de pruebas.

Causa de los defectos

El software es elaborado por seres humanos que pueden cometer errores que dan lugar a defectos. Estos defectos pueden generar un fallo.

Las causas de los errores pueden ser: presión en los tiempos, complejidad de la aplicación o la arquitectura, tecnologías cambiantes, existencia de un gran número de interfaces, etc…

Al margen de la existencia de un defecto, pueden producirse fallos por condiciones ambientales (radiación, magnetismo, campos eléctricos, etc…)

Calidad del software

Características que debe cumplir una funcionalidad:

- Idoneidad
- Precisión
- Conformidad
- Interoperabilidad
- Seguridad

Características que debe cumplir un requisito no funcional:

- Fiabilidad
- Usabilidad
- Eficiencia
- Mantenibilidad
- Portabilidad

Depuración vs Prueba

La depuración es la actividad de desarrollo que localiza, analiza y elimina las causas de los fallos. Normalmente, la responsabilidad de las pruebas la tienen los probadores, mientras que los desarrolladores suelen tener la de la depuración (debugging)

Siete principios de las Pruebas

1. **Las pruebas muestran la presencia de defectos**, no la ausencia de ellos.
2. **Las pruebas exhaustivas no son posibles**, ya que el coste aumenta exponencialmente.
3. **Pruebas tempranas**. Cuanto antes se descubra un defecto, menos costosa será su corrección.
4. **Bloques de defectos**. Unos pocos módulos contienen la mayor parte de los defectos descubiertos durante la fase de pruebas o son responsables de la mayoría de los fallos en producción.
5. **La paradoja del pesticida**. Si se repiten una y otra vez los casos de prueba, se llegará a que el mismo conjunto de casos de prueba no sirva para localizar nuevos defectos. Para superar esta "paradoja del pesticida", los casos de prueba necesitan ser revisados de manera regular y se necesita escribir casos nuevos para ejercitar diferentes partes del software o sistema para localizar más defectos.
6. **Las pruebas dependen del contexto**. Las pruebas se llevan a cabo de manera diferente en contextos diferentes. Por ejemplo, el software crítico desde el punto de vista de la seguridad de las personas se prueba de forma distinta que la seguridad de un portal de comercio electrónico.
7. **La falacia de la ausencia de errores**. Localizar y corregir defectos no sirve de nada si el sistema construido no cubre las necesidades y expectativas de los usuarios.

Proceso de pruebas

Planificación

Principales tareas:

- Determinar el alcance y los riesgos.
- Identificar los objetivos de las pruebas y los criterios de finalización.
- Seleccionar y priorizar las pruebas.
- Determinar las técnicas, coberturas, herramientas, entorno y equipos de prueba.
- Determinar los métodos y estrategia.
- Planificar las actividades.
- Adquirir/planificar los recursos, Personal, Entorno y medios de apoyo, Coste.

Control

Principales tareas:

- Tomar decisiones basadas en la información que le proporciona la monitorización de las pruebas.
- Re Priorizar las pruebas cuando aparece un riesgo identificado (por ejemplo, retraso en la entrega del software).
- Cambiar la planificación según la disponibilidad del entorno.

Análisis y diseño de las pruebas

Principales tareas:

- Analizar la documentación en base a la cual se van a generar los casos de prueba.
- Evaluar la capacidad de poder probar los objetos de prueba.
- Diseñar y priorizar casos de prueba lógicos.
- Identificar los datos de prueba.
- Establecer las precondiciones para poder ejecutar las pruebas.
- Diseñar/adaptar el entorno de pruebas.
- Definir la operación del entorno de prueba, incluyendo la administración de usuarios.
- Crear trazabilidad bidireccional entre la documentación base y los casos de prueba.

Implementación de las pruebas

Principales tareas:

- Desarrollo, implementación y priorización de casos de prueba.
- En su caso, escribir scripts de pruebas automatizadas.
- Crear secuencias de prueba para mejorar la eficiencia de la ejecución.
- Implementar controladores de pruebas.
- Implementar/configurar/verificar el entorno de pruebas.
- Instalar herramientas.
- Integrar los datos de prueba.
- Verificar y actualizar la trazabilidad bidireccional entre la documentación base y los casos de prueba.

Ejecución de las pruebas

Principales tareas:

- Ejecutar los casos de prueba.
- Registrar y analizar resultados, Objeto de prueba, Probador, Datos de prueba, Resultado.
- Comparar los resultados obtenidos con los resultados esperados e informar acerca de las discrepancias, analizándolas. (Puede generar defecto)
- Repetir las actividades de prueba según la acción que se lleve a cabo para cada discrepancia.

Evaluación y documentación de las pruebas

Principales tareas:

- La evaluación de los criterios de finalización es la actividad en la que se valora la ejecución de la prueba respecto de los objetivos definidos. Debería llevarse a cabo en cada nivel de prueba que implica.
- Si no se cumplen los criterios, habrá que comprobar si realmente es posible cumplirlos y si la planificación de las pruebas tiene que ser adaptada.
- Los errores encontrados llevan a un nuevo ciclo de pruebas.
- Se ha de proporcionar información suficiente para ayudar a la decisión de los casos de prueba.
- Se ha de generar un informe final de pruebas dirigido a las áreas afectadas.

Criterios de finalización de las pruebas

Principales tareas:

- Ratio de localización de errores.
- Finalización de las pruebas por motivos de coste o tiempo.
- Cobertura de código.
- Cobertura de riesgos.

Actividades de cierre

Principales tareas:

- Recoger datos de las actividades de prueba completadas para consolidar experiencia.
- Cierre de los informes de incidencias o apertura de solicitudes de cambio para todos los puntos que sigan abiertos.
- Comprobación de que entregables han sido efectivamente entregados y aprobados.
- Documentación de la aceptación del sistema.
- Archivado de las pruebas, el entorno de pruebas y la infraestructura de pruebas para posterior reutilización.
- Análisis de las lecciones aprendidas para futuros proyectos.
- Usar la información recopilada para mejorar la madurez de las pruebas.

Psicología de Pruebas

Diferencias entre diseñar, desarrollar y probar

Las pruebas requieren una forma diferente de pensar de las que aquellos que diseñan o desarrollan sistemas:

- La misión del diseñador de pruebas es ayudar al cliente suministrándole los requisitos correctos.
- La misión del desarrollador es convertir los requisitos en funciones.
- La misión del probador es examinar la correcta implementación de los requisitos del usuario.
- Objetivo común: Proporcionar buen software.

Mentalidad tradicional

Desarrollador	Probador
Transforma los requisitos	Planifica sus pruebas
Desarrolla estructuras	Especifica casos de prueba
Programa software	Sólo quiere encontrar errores
Consigue un producto	Los errores del programador son su éxito

El trabajo del desarrollador es constructivo, mientras que el del probador es destructivo

¡COMPLETAMENTE FALSO!

Las pruebas también son constructivas, ya que su objetivo es un producto libre de errores.

Pruebas del desarrollador

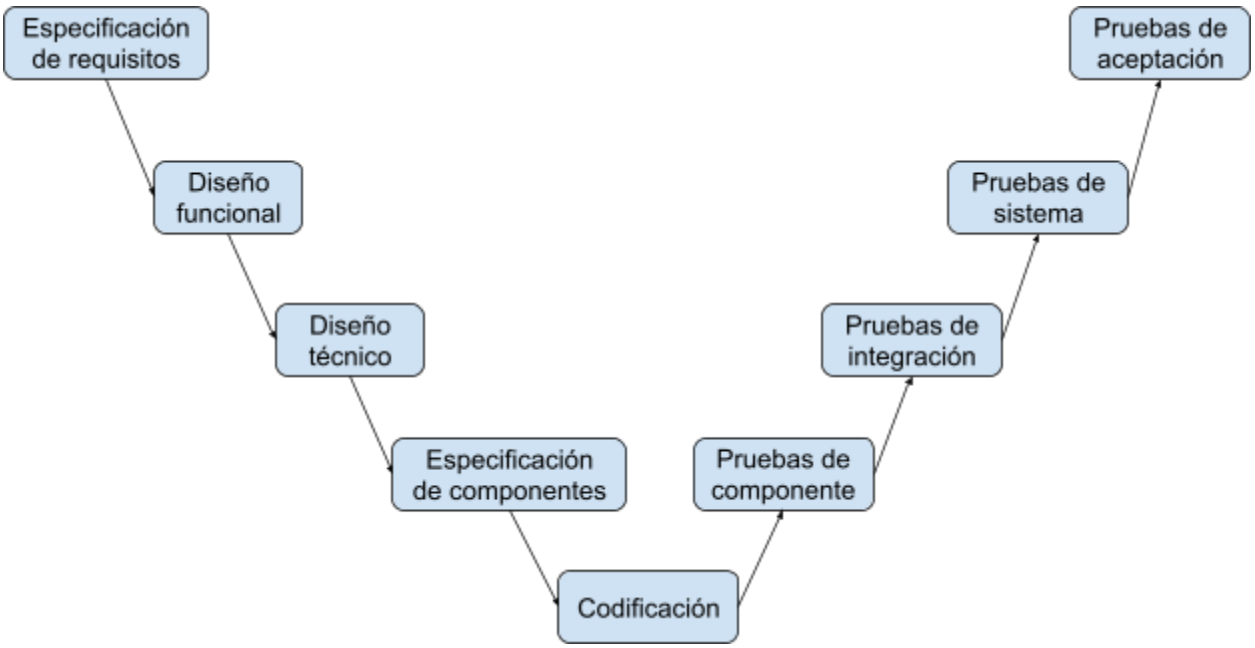
- El desarrollador nunca se mantiene neutral respecto de su “creación”.
- El ser humano tiende a ser ciego respecto a sus errores.
- Los errores como consecuencia de una mala interpretación de los requisitos quedan descubiertos.

Personalidad de un buen probador

- Curioso, atento a los detalles.
- Escepticismo y ojo crítico.
- Buenas capacidades de comunicación.
- La comunicación positiva puede ayudar a evitar o facilitar situaciones complicadas.
- Tiene experiencia en el sector.

Las pruebas en el ciclo de vida del software

Modelo tradicional del desarrollo de software en V o en cascada.



Fases del desarrollo (Rama izquierda)

- **Especificación de requisitos**, Historia de Usuario.
- **Diseño funcional**, Diagramas de Flujo.
- **Diseño técnico**, UML Diagrama de Clases.
- **Especificación de componentes**, Interfaces de Java.
- **Codificación**, Clase de Java.

Fase de Pruebas (Rama derecha)

Pruebas de componentes

Pruebas de cada uno de los componentes básicos del software respecto de su programación. Debido a las diferentes definiciones de componentes básicos de software en los distintos lenguajes de programación existen diferentes definiciones para las pruebas de componentes:

- Pruebas de módulo (lenguajes como C)
- Pruebas de clases (lenguajes como Java)
- Pruebas de unidad (lenguajes con Pascal)

Los componentes comprobados serán, respectivamente, módulos, clases o unidades.

¿Qué se prueba?

- Pruebas completas de los componentes individuales
- Cada componente se prueba de manera separada y aislada.

¿Por qué se prueba?

- ¿Se cumplen todas las especificaciones?
- ¿Se ejecutan correctamente todas las funciones?
- ¿Cada una de las funciones de un componente se comprueban al menos con un caso de prueba?

Pruebas de integración

Pruebas de interrelación de los elementos del software tras su integración. La integración es el proceso por el cual los elementos básicos del software son agrupados en elementos mayores. Este tipo de pruebas analizan la interoperabilidad entre diferentes componentes de un software.

¿Qué se prueba?

En las pruebas de integración se comprueba la interoperabilidad entre elementos básicos del software. Cada elemento básico debería haber sido comprobado antes de que comience la integración.

¿Por qué se prueba?

El objetivo de las pruebas de integración radica en la búsqueda de posibles errores de interfaces, se comprueba el correcto funcionamiento de los componentes, sobretodo con vistas a los siguientes niveles de pruebas como pueden ser las pruebas de rendimiento o de seguridad.

Pruebas del sistema

Comprobación del sistema integrado completo respecto del cumplimiento de los requisitos específicos. Desde el punto de vista técnico, ya se han probado todos los componentes y su interrelación, faltan las pruebas del sistema completo en condiciones de funcionamiento y desde el punto de vista del usuario, como el entorno, funciones, carga, etc…

¿Qué se prueba?

Pruebas del sistema integrado desde el punto de vista del usuario. El entorno de pruebas debería asemejarse al entorno de producción pero sin realizar pruebas en dicho entorno

¡NUNCA!

Estas pruebas deben validar el cumplimiento de los requisitos establecidos, por lo que debemos categorizarlos por:

- Pruebas funcionales, son aquellas pruebas que verifican el correcto funcionamiento a nivel de los requisitos definidos previos al desarrollo.
- Pruebas no funcionales, tales como pruebas de Fiabilidad, Usabilidad, Eficiencia, Mantenibilidad y Portabilidad.

Pasamos a detallar más este tipo de pruebas en cada uno de los grupos.

Pruebas funcionales

- Pruebas basadas en riesgos y/o especificaciones de requisitos.
- Pruebas basadas en procesos de negocio.
- Pruebas basadas en casos de uso.

Pruebas no funcionales

- **Pruebas de carga:** ¿cómo se comporta el sistema bajo condiciones de carga nominales?
- **Pruebas de rendimiento:** ¿cómo de rápido es el sistema en determinadas funciones/casos de uso?
- **Pruebas de volumen:** ¿cómo se comporta el sistema al procesar grandes cantidades de conjuntos de datos/archivos?
- **Pruebas de estrés:** ¿cómo reacciona el sistema con sobrecarga?, ¿cómo reacciona el sistema al regresar al modo de funcionamiento normal?
- **Pruebas de seguridad:** ¿Está protegido el sistema frente a accesos no autorizados?, ¿están protegidos los datos frente a accesos no autorizados o pérdidas?
- **Pruebas de estabilidad:** ¿con qué frecuencia se cae el sistema por unidad de tiempo determinada?, comportamiento del sistema en funcionamiento prolongado.
- **Pruebas de robustez:** ¿cómo reacciona el sistema frente a una utilización incorrecta o frente a errores de entrada?, comportamiento del sistema frente a defectos de hardware y el regreso al funcionamiento normal.

Pruebas de aceptación

Las pruebas de aceptación comprueban el producto desde el punto de vista del usuario o del cliente antes de su paso a producción, ¿se cumplen las expectativas del usuario/cliente?. El usuario/cliente está involucrado, dependiendo del grado, en la personalización del software.

Estas pruebas también validan la aceptación del Contrato y Normativa que deba cumplirse para su correcto funcionamiento. Los criterios de aceptación han de quedar definidos, de forma clara y explícita, en el momento en que ambas partes acuerdan el contrato.

¿Qué se prueba?

Pruebas de aceptación del Contrato y Normativa: se rigen por los Criterios de Aceptación que se refieren a cualquier normativa a la que tenga que adherirse el Contrato, como pueden ser gubernamentales, legales o de seguridad.

- **Pruebas de aceptación del usuario:** ¿se acepta el software como parte de los usuarios últimos?. Las pruebas de aceptación deben realizarse en función de la personalización del software.
- **Pruebas de aceptación operacional:** Aceptación del sistema por parte de los Administradores. Incluyen:
    - Pruebas de backup/restore
    - Recuperación de desastres
    - Gestión de usuarios
    - Tareas de mantenimiento
    - Tareas de carga y migración de datos
    - Comprobación periódica de vulnerabilidades de seguridad
- **Pruebas de campo (alfa y beta):** Distribución anticipada de versiones estables del software a una selección representativa del círculo de clientes. Alternativamente primero pruebas a nivel alfa de una versión previa y posteriormente beta. Estas pruebas deben asegurar más allá de las pruebas del sistema, que el software se puede ejecutar en un gran número de entornos diferentes.

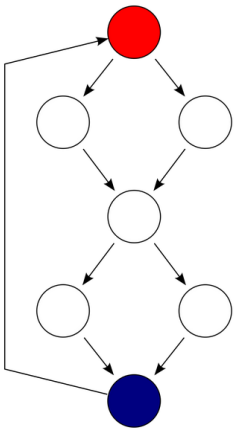
Pruebas de regresión

Este tipo de pruebas es un subconjunto de las pruebas realizadas previamente y que puede contener cualquiera de ellas. Estas pruebas destacan debido a que pueden ser ejecutadas cada vez que se hace una nueva iteración de cambios del proyecto para reducir el riesgo de que aparezcan defectos pero con un coste en ejecución de pruebas menor, sobre todo si estas pruebas están automatizadas.

Complejidad ciclomática

La complejidad ciclomática es el cálculo que identifica la cantidad de rutas que pueden aparecer en un flujo funcional.

Dado el siguiente caso:



Calcular la complejidad ciclomática para el flujo desde el punto rojo (inicio) hasta el punto azul (final).

Podemos apreciar que se tienen **4** rutas diferentes dentro del flujo funcional reflejado.

Fase de pruebas en proyectos Agile

Beneficios de enfoques ágiles frente enfoques tradicionales

Para saber los beneficios del enfoque ágil frente al de los tradicionales, se deben comprender las diferencias entre las pruebas en los modelos de vida tradicionales y los ciclos de vida ágiles para poder trabajar con eficacia y eficiencia. Los modelos ágiles difieren en cuanto a la forma en que se integran las actividades de prueba y desarrollo, los productos de trabajo del proyecto, los nombres, los criterios de entrada y salida utilizados para los distintos niveles de prueba, el uso de herramientas y la forma en que se pueden utilizar eficazmente la prueba independiente.

Los probadores deben recordar que las organizaciones varían considerablemente en su aplicación de los ciclos de vida.

Una de las principales diferencias entre los ciclos de vida tradicionales y los ciclos de vida ágiles es la idea de iteraciones muy cortas, cada una de las cuales da como resultado un software funcionando que ofrece prestaciones de valor a los implicados del negocio.

Los probadores, los desarrolladores y los implicados del negocio tienen ***todos*** un papel en la prueba, como en los ciclos de vida tradicionales.

En algunas prácticas ágiles, se utiliza el trabajo en pareja.

La automatización de la prueba en todos los niveles tiene lugar en muchos equipos ágiles, creados tanto por desarrolladores (pruebas unitarias y de integración) como probadores (pruebas funcionales).

Un principio ágil fundamental es que el cambio puede producirse a lo largo del proyecto. Por lo tanto, en los proyectos ágiles se favorece la documentación ligera del producto de trabajo. Los cambios en las prestaciones existentes tienen implicaciones en la prueba, especialmente en la prueba de regresión. El uso de pruebas automatizadas es una forma de gestionar la cantidad de esfuerzo de prueba asociado al cambio.

Las historias de usuario son la forma ágil de especificación de requisitos y deben explicar cómo debe comportarse el sistema con respecto a una característica o función única y coherente.

Los productos de trabajo típicos de los desarrolladores en los proyectos ágiles incluyen el código. Los desarrolladores ágiles también suelen crear pruebas unitarias automatizadas. Estas pruebas pueden crearse tras el desarrollo del código. Sin embargo, en algunos casos, los desarrolladores crean pruebas de forma incremental, antes de que se escriba cada fragmento de código, con el fin de proporcionar una forma de verificar, una vez escrita esa porción de código.

Los productos de trabajo típicos de los probadores en los proyectos ágiles incluyen pruebas automatizadas, así como documentos, como planes de prueba, catálogos de riesgos de calidad, pruebas manuales, informes de defectos y registros de resultados de pruebas.

Uno de los objetivos de las pruebas automatizadas es confirmar que la construcción funciona y es instalable. Esto requiere una inversión en informes de prueba en tiempo real para proporcionar una buena visibilidad de los resultados de la prueba.

Las herramientas de construcción y pruebas automatizadas ayudan a gestionar el riesgo de regresión asociado a los frecuentes cambios que suelen producirse en los proyectos ágiles. Sin embargo, confiar demasiado en las pruebas unitarias automatizadas para gestionar estos riesgos puede ser un problema, ya que las pruebas unitarias suelen tener una efectividad limitada en la detección de defectos. También son necesarias las pruebas automatizadas a nivel de integración y de sistema.

Rol y Competencias de un Probador en un Equipo Ágil

En un equipo ágil, los probadores deben colaborar estrechamente con todos los demás miembros del equipo y con los implicados del negocio. Esto tiene una serie de implicaciones en cuanto a las competencias que debe tener un probador y las actividades que realiza dentro de un equipo ágil.

Los probadores ágiles deben tener todas las competencias mencionadas anteriormente, además de estás competencias, un probador en un equipo ágil debe ser competente en la automatización de la prueba, el desarrollo guiado por pruebas, la prueba de aceptación, caja blanca caja negra y la prueba basada en la experiencia.

Dado que las metodologías ágiles dependen en gran medida de la colaboración, la comunicación y la interacción entre los miembros del equipo y los implicados fuera de él, los probadores de un equipo ágil deben tener buenas competencias interpersonales. Los probadores de los equipos ágiles deberían:

- 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 las mismas y la calidad del producto.
- Trabajar eficazmente para definir historias de usuario que puedan ser probadas, especialmente los criterios de aceptación, con los representantes del cliente e implicados.
- 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.

El crecimiento continuo de las competencias, incluido el de las habilidades interpersonales, es esencial para todos los probadores, incluidos los de los equipos ágiles.

El papel de un probador en un equipo ágil incluye actividades que generan y proporcionan retroalimentación no sólo sobre el estado de la prueba, el avance de la misma y la calidad del producto, sino también sobre la calidad del proceso. Además de las actividades descritas en otras partes de este programa de estudio, estas actividades incluyen:

- 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.
- Entrenar a otros miembros del equipo en los aspectos relevantes de la prueba.
- Asegurar que se programen las tareas de prueba adecuadas durante la planificación de la entrega y la iteración.
- Colaborar activamente con los desarrolladores y los implicados del negocio para aclarar los requisitos, especialmente en términos de capacidad de ser probado, consistencia y completitud.
- Participar de forma proactiva en las retrospectivas del equipo, sugiriendo e implementando mejoras.

Dentro de un equipo ágil, cada miembro del equipo es responsable de la calidad del producto y desempeña un papel en la realización de tareas relacionadas con la prueba.Las organizaciones ágiles pueden encontrar algunos riesgos organizativos relacionados con la prueba:

- Los probadores trabajan tan estrechamente con los desarrolladores que pierden la mentalidad de probador adecuada.
- Los probadores se vuelven tolerantes o guardan silencio sobre las prácticas ineficientes, ineficaces o de baja calidad dentro del equipo.
- Los probadores no pueden seguir el ritmo de los cambios que se producen en las iteraciones con limitaciones de tiempo.

Métodos de Prueba Ágil

Desarrollo Guiado por Pruebas

El desarrollo guiado por pruebas (TDD, Test-Driven Development) se utiliza para desarrollar código guiado por casos de prueba automatizados. El proceso para el desarrollo guiado por pruebas es:

- Añadir 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 vuelva 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.

Las pruebas escritas son principalmente de nivel unitario y se centran en el código, aunque también pueden escribirse pruebas a nivel de integración o de sistema.

Desarrollo Guiado por Pruebas de Aceptación

El desarrollo guiado por pruebas de aceptación define los criterios de aceptación y las pruebas durante la creación de las historias de usuario. El desarrollador guiado por pruebas de aceptación 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 del negocio para garantizar este comportamiento.

El desarrollo guiado por pruebas de aceptación crea pruebas reutilizables para las pruebas de regresión.

Desarrollo Guiado por el Comportamiento

El desarrollo guiado por el comportamiento permite al desarrollador concentrarse en probar el código basándose en el comportamiento esperado del software.

Pueden utilizarse marcos de desarrollo guiados por el comportamiento para definir criterios de aceptación basados en el formato dado/cuando/entonces:

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

A partir de estos requisitos, el marco de desarrollo guiado por comportamientos genera un código que puede ser utilizado por los desarrolladores para crear casos de prueba. El desarrollo guiado por el comportamiento ayuda al desarrollador a colaborar con otras partes interesadas, incluidos los probadores, para definir pruebas unitarias precisas centradas en las necesidades del negocio.

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 y aceptación. La pirámide de prueba hace hincapié en tener un gran número de pruebas en los niveles inferiores y, a medida que el desarrollo avanza hacia los niveles superiores, el número de pruebas disminuye. Por lo general, las pruebas unitarias y de nivel de integración se automatizan y se crean utilizando herramientas basadas en API. En los niveles de sistema y aceptación, las pruebas automatizadas se crean utilizando herramientas basadas en los criterios de aceptación. El concepto de pirámide de prueba se basa en el principio de control de calidad y pruebas tempranas.

Cuadrante de Pruebas

Los cuadrantes de prueba, alinean los niveles de prueba con los tipos de prueba adecuados en la metodología. El modelo de cuadrantes de prueba, y sus variantes, ayuda a garantizar que todos los tipos de niveles de prueba importantes se incluyen en el ciclo de vida de desarrollo. Este modelo también proporciona una forma de diferenciar y describir los tipos de pruebas a todos los implicados, incluidos los desarrolladores, probadores y representantes de negocio.

En los cuadrantes de prueba, éstas pueden estar orientadas al negocio (usuario) o a la tecnología (desarrollador).

- El cuadrante Q1 es el nivel unitario, está orientado a la tecnología y apoya a los desarrolladores. Este cuadrante contiene pruebas unitarias. Estas pruebas deben automatizarse e incluirse en el proceso de integración continua.
- El cuadrante Q2 es el nivel sistema, de cara al negocio, y confirma el comportamiento del producto. Este cuadrante contiene pruebas funcionales, ejemplos, pruebas de historia, prototipos de experiencia de usuario y simulaciones. Estas pruebas comprueban los criterios de aceptación y pueden ser manuales o automatizadas. Suelen crearse durante el desarrollo de la historia de usuario y así mejoran la calidad de las historias. Son útiles para crear conjuntos de pruebas de regresión automatizadas.
- El cuadrante Q3 es el nivel de aceptación de sistema o usuario, de cara al negocio, y contiene pruebas que critican el producto, utilizando escenarios y datos realistas. Este cuadrante contiene pruebas exploratorias, escenarios, flujos de procesos, pruebas de usabilidad, pruebas de aceptación de usuario, pruebas alfa y pruebas betas. Estas pruebas suelen ser manuales y están orientadas al usuario final.
- El cuadrante Q4 es el nivel de aceptación de sistema u operativa, orientado a la tecnología y contiene pruebas que critican el producto. Este cuadrante contiene pruebas de rendimiento, carga, estrés y escalabilidad, pruebas de seguridad, mantenibilidad, gestión de la memoria, compatibilidad e interoperabilidad, migración de datos, infraestructura y pruebas de recuperación. Estas pruebas suelen estar automatizadas.

Durante cualquier iteración, pueden requerirse pruebas de alguno o de todos los cuadrantes. Los cuadrantes de prueba se aplican a las pruebas dinámicas más que a las pruebas estáticas.