Testing ágil

Tradicional vs. ágil

Testing tradicional → se realiza al final, es una etapa definida. Es organizado por un líder de proyecto, la colaboración con el cliente se realiza al inicio. Se ofrece una cobertura exhaustiva de software y una cobertura máxima de defectos. Requiere tiempo y esfuerzos, hay poca interacción entre los testers.

Vs.

Testing Ágil → enfoque colaborativo, durante todo el proyecto. La colaboración con el cliente es activa, al final de cada etapa de prueba. Permite ajustarse a requerimientos cambiantes del cliente, asegura una entrega rápida sin perder calidad. Es un desafío en los proyectos extensos, difícil de mantener por la ausencia de una documentación exhaustiva y de saber cuándo se terminó de testear todo.

Pirámide del testing

Es una metáfora que sugiere agrupar las pruebas según la granularidad. Permite disminuir el número de pruebas manuales.

En la base están las pruebas unitarias, generadas por el mismo desarrollador. Son fácilmente automatizables. El mayor esfuerzo debería venir acá (70%), porque son las pruebas más sólidas y que permiten la detección más temprana de errores.

En el medio están las pruebas de servicio, donde se ven cómo los componentes se tratan entre sí (pruebas de integración y de contrato). Hay un menor volumen de pruebas, pero son más complejas. Se recomienda un esfuerzo de 20%.

En la cima están las pruebas de interfaz de usuario, donde se ejecuta la funcionalidad completa. Se recomienda un esfuerzo de 10%, intentando no gastar demasiado tiempo en estas pruebas que son fáciles de derrumbar.

Sobre la pirámide hay una nube que son las pruebas exploratorias, que sirven para construir las clases de pruebas.

En un enfoque tradicional, la pirámide se invierte.

Prácticas de testing ágil

Herramientas técnicas para el desarrollo de pruebas que siguen los principios del desarrollo ágil. Ponen tanto el foco en el cliente como en el desarrollo.

Desarrollo Dirigido por Pruebas → entrelaza el desarrollo de las pruebas y el código. Se hacen las pruebas. Se desarrolla el código y las pruebas de forma incremental: cada pieza de código tendrá al menos una prueba.

Desarrollo Dirigido a Pruebas de Aceptación → se busca una comprensión compartida de los requerimientos del sistema. Se hacen especificaciones en lenguaje natural y luego pruebas de aceptación automatizadas para probar el sistema. Evita el desarrollo de funcionalidades inútiles o que no entregan valor.

Pruebas exploratorias → el tester va probando el sistema de forma no estructurada. Se utiliza en casos donde se requiere aprender rápido el producto o analizarlo desde la perspectiva del usuario.

Pruebas de unidad o de integración automatizadas → se generan las pruebas en código (scripts de prueba) y luego se corre ese código sobre esas pruebas.

Control de versión de pruebas con el código → se prueban las distintas funcionalidades en sus distintas versiones.

Pruebas de regresión → proceso en el que se evalúa todo el sistema, incluso aquellas funciones donde no se han producido cambios, para poder verificar que el sistema sigue funcionando de forma correcta.

Cuadrantes del testing

Forma de organizar las pruebas para adaptarlas a entornos ágiles, con orientación al negocio, tecnologías y colaboración.

Primer cuadrante: Da soporte al equipo desde una perspectiva de la tecnología. Se busca la calidad del software. Pruebas unitarias, TDD, integración continua.

Segundo cuadrante: Perspectiva del negocio. Se busca la calidad externa del sistema. User stories y criterios de aceptación.

Tercer cuadrante: Perspectiva del usuario. Se valida que el producto haga lo que tiene que hacer. Pruebas exploratorias.

Cuarto cuadrante: Perspectiva no funcional. Pruebas de rendimiento: seguridad, velocidad, integridad. Se busca observar qué tan robusto es el sistema.

Roles y competencias del tester

Un tester es aquella persona que realiza pruebas de software durante el desarrollo antes de que llegue al usuario final, realizando documentación sobre las pruebas realizadas

Consultor; profesional con experiencia y conocimiento, asesora de estrategias y mejores técnicas

Tester: define los casos de prueba y los ejecuta teniendo como resultado los informes

Enfoque tradicional: se planifica antes de hacer el testing y se hace una documentación exhaustiva, se realiza en procesos definidos por lo que los requerimientos no cambian

Enfoque ágil: integración continua del testing y desarrollo, asegura la calidad, automatiza los test

Competencias: conocimientos técnicos, habilidades de comunicación, ser hábil solucionador de problemas, tener mirada tanto de detalle como general.

Principios y valores. Retroalimentación continua teniendo feedback con el cliente, entregar valor al cliente, comunicación cara a cara, coraje (se aprende de las fallas), mantenerlo simple haciendo lo justo y necesario, responder al cambio (aceptamos que ellos cambios pueden ocurrir) autoorganizarse, foco en las personas, disfrutar el proceso ya que es algo colaborativo

Las competencias entre ágil y tradicional son lo mismo.

Desafíos culturales

Sacrispin, los equipos de prueba son los que mas demoran en hacer esta transición ya que se han arraigado porque se encuentran más aislados.

Cultura organizacional: valores, formas.

Hipótesis: no identificar y abordar desafíos culturales es un factor determinante del fracaso.

Perdida de identidad de los testers: se arraigan los testers a el área de testing por miedo al cambio.

Desafío de cambiar la mentalidad del tester de policía que busca errores y culpa a los programadores.

Resistencia al desarrollo sostenible: en tradicional se hace el testing al final y esto va en contra de los principios agiles

Principales motivos de fracasos en proyectos agiles: 4 de los top 5 son desafíos culturales

Falta de experiencia con métodos agiles, cultura de la empresa en desacuerdo con valores agiles fundamentales, falta de apoyo gerencial, presión externa para seguir los procesos tradicionales en cascada, falta de apoyo a la transición cultural,

No solo se deben tener en cuenta cambios técnicos ante los cambios organizacionales sino también se deben tener consideración sobre los cambios culturales

Soluciones: un lugar donde hablar de su miedo, celebrar todos los éxitos por pequeños que sean, contratar un especialista en pruebas agiles como una especie de mentor para ayudar a que se integren en la cultura ágil

Manifiesto de testing

Es un conjunto de principios y valore que se encargan de guiar hacia el éxito la actividad del testing

Testing durante todo el proceso sobre testing al final: es un error tener el testing al final del desarrollo, se debe establecer durante todo el proceso de desarrollo de software, el testing es lo que permite mejorar la calidad del producto

Prevenir bugs sobre encontrarlos: en enfoques tradicionales se mide la efectividad según la cantidad de bugs encontrados, en cambio ágil intenta tratar de evitar bugs, ya que estos suelen surgir por falta de conocimiento y no preguntar por lo que todos deben tener una misma base de conocimiento

Entender lo que se prueba sobre verificar la funcionalidad: se debe entender lo que quiere el cliente para que el producto sea de la calidad que se busca

Construir el mejor sistema sobre destruirlo: en tradicional el testing y el desarrollo parecen enemigos y están aparte, en cambio en ágil se plantea una relación más cercana entre ambos roles

Responsabilidad del equipo para la calidad sobre responsabilidad del tester: la calidad es responsabilidad compartida del equipo y no solo cae sobre el tester

Automatización de testing

Beneficios: mejor detección de incidentes porque se pueden realizar mayor cantidad de pruebas, mayor cobertura de pruebas, reducción de costo

Herramientas: nunit, selenium, appium

Pirámide de Mike conin: plantea como estructurar las pruebas, como base pone a las pruebas unitarias ya que son las más baratas, fáciles y rápidas de ejecutar y evita que los defectos se propaguen mas arriba en el proceso de desarrollo.

Un mito común es que se puede automatizar todo, se conviene automatizar los puntos mas críticos para identificar con mayor velocidad los errores y en los puntos mas complejos ya que un tester manual es mas propenso a cometer un error al momento de testear