Unidad 3
Métodos, técnicas y herramientas de pruebas ágiles

1. Métodos de pruebas ágiles

Existen determinadas prácticas de pruebas que pueden seguirse en cualquier proyecto de desarrollo (ágil o no) para producir productos de calidad. Entre ellas se encuentra escribir pruebas con antelación para expresar el comportamiento correcto, centrarse en la prevención, detección y eliminación temprana de defectos, y asegurar que se ejecutan los tipos de pruebas adecuados en el momento idóneo y como parte del nivel de prueba correcto. Los profesionales Ágiles buscan introducir estas prácticas cuanto antes. Los probadores en proyectos ágiles desempeñan una función clave al orientar sobre el uso de estas prácticas de pruebas durante todo el ciclo de vida.

1.1 Desarrollo guiado por pruebas, desarrollo guiado por pruebas de aceptación y desarrollo guiado por el comportamiento

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 los equipos ágiles para llevar a cabo las pruebas en los distintos niveles de pruebas. Cada técnica es ejemplo de un principio fundamental de las pruebas: el beneficio de las pruebas tempranas y de las actividades de aseguramiento de la calidad, ya que las pruebas se definen antes de escribir el código.

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 del desarrollo guiado por pruebas es:
- Añadir una prueba que capte el concepto que tiene el programador del funcionamiento deseado de un pequeño trozo de código.
- Ejecutar la prueba, que debería fallar ya que el código no existe.
- Escribir el código y ejecutar la prueba hasta que se pase la prueba.
- Refactorizar el código una vez pasada la prueba, volver a ejecutar la prueba para asegurar que sigue pasando contra el código refactorizado.
- Repetir este proceso para el siguiente pequeño trozo de código, ejecutando tanto

las pruebas previas como las pruebas añadidas.

Las pruebas escritas son principalmente de nivel unitario y están centradas en el código, si bien también pueden escribirse pruebas en los niveles de integración o sistema. El desarrollo guiado por pruebas se hizo popular gracias a la Programación Extrema [Beck02], pero también se utiliza en otras metodologías ágiles y a veces en ciclos de vida secuenciales. Este tipo de desarrollo ayuda a los desarrolladores a concentrarse en resultados esperados claramente definidos. Las pruebas se automatizan y se utilizan en la integración continua.

Desarrollo guiado por pruebas de aceptación

El desarrollo guiado por pruebas de aceptación [Adzic09] define los criterios y las pruebas de aceptación durante la creación de historias de usuario .El desarrollo guiado por pruebas de aceptación es un enfoque colaborativo que permite a todas las partes interesadas conocer cómo tiene que comportarse el componente de software y qué necesitan los desarrolladores, probadores y representantes de negocio para asegurar este comportamiento.
El desarrollo guiado por pruebas de aceptación crea pruebas reutilizables para las pruebas de regresión. Existen herramientas específicas que ayudan a la creación y la ejecución de dichas pruebas, a menudo dentro del proceso de integración continua. Estas herramientas pueden conectarse a capas de datos y servicio de la aplicación, lo que permite ejecutar pruebas a nivel de sistema o aceptación. El desarrollo guiado por pruebas de aceptación permite la rápida resolución de defectos y la validación del comportamiento de las funcionalidades. Ayuda a determinar si se cumplen los criterios de aceptación para la funcionalidad.

Desarrollo guiado por el comportamiento

El desarrollo guiado por el comportamiento [Chelimsky10] permite al desarrollador centrarse en probar el código basándose en el comportamiento esperado del software. Normalmente las pruebas son más fáciles de entender para los demás miembros del equipo y partes interesadas dado que se basan en el comportamiento manifestado del software.
Los marcos de trabajo del desarrollo guiado por el comportamiento pueden utilizarse para definir criterios de aceptación tomando como base el formato dado/cuando/entonces:

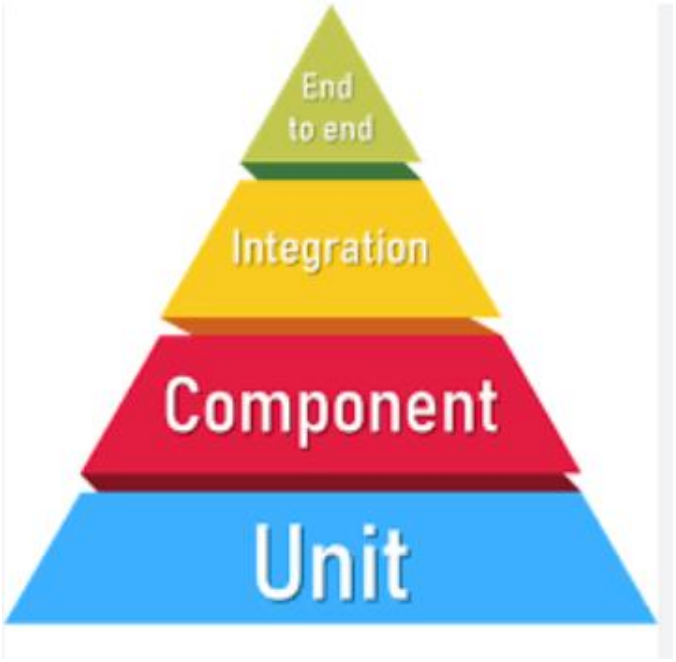
- *Dado* un contexto inicial
- *Cuando* se da un evento
- *Entonces* se verifican ciertos resultados.

A partir de estos requisitos, el marco de trabajo del desarrollo guiado por el comportamiento genera código que los desarrolladores pueden utilizar para crear casos de prueba. El desarrollo guiado por el comportamiento ayuda al desarrollador a colaborar con otras partes interesadas, incluyendo probadores, para definir pruebas unitarias precisas centradas en las

necesidades de negocio.

1.2 La pirámide de pruebas

Un sistema de software puede probarse en distintos niveles. Los típicos niveles de pruebas son, desde la base de la pirámide hasta lo más alto, nivel unitario, de integración, de sistema y de aceptación (remítase a [ISTQB_FL_SYL], Sección 2.2). La pirámide de pruebas destaca por tener un número mayor 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 se reduce (vértice de la pirámide). Normalmente las pruebas de nivel unitario y de integración se automatizan y se crean empleando herramientas basadas en API. En los niveles de sistema y aceptación, las pruebas automatizadas se crean empleando herramientas basadas en GUI. El concepto de la pirámide de pruebas se basa en el principio de aseguramiento de la calidad temprano y pruebas tempranas (es decir, eliminando los defectos lo antes posible en el ciclo de vida).



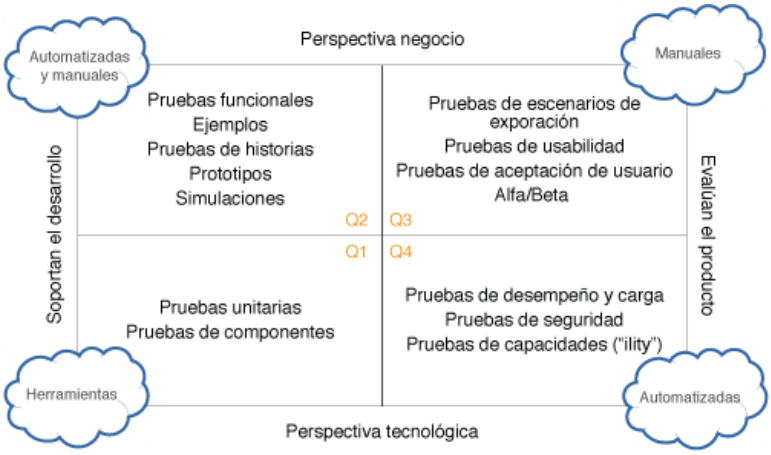
1.3 Cuadrantes de pruebas, niveles de pruebas y tipos de pruebas

Los cuadrantes de pruebas, definidos por Brian Marick [Crispin08], alinean los niveles de pruebas con los tipos de pruebas adecuados en la metodología ágil. El modelo de cuadrantes de pruebas, y sus variantes, ayuda a garantizar que todos los tipos de pruebas y niveles de pruebas importantes se incluyen en el ciclo de vida del desarrollo. Este modelo también constituye una forma de diferenciar y describir los tipos de pruebas para todas las partes interesadas, incluyendo desarrolladores, probadores y representantes de negocio.

En los cuadrantes de pruebas, las pruebas pueden referirse al negocio (usuario) o la tecnología (desarrollador). Algunas pruebas soportan el trabajo realizado por el equipo ágil y confirman el comportamiento del software. Otras pruebas pueden verificar el producto. Las pruebas pueden ser totalmente manuales, estar totalmente automatizadas, ser una combinación de pruebas manuales y automatizadas o ser manuales pero estar soportadas por herramientas. Los cuatro cuadrantes son los siguientes:

- *El cuadrante Q1* es el nivel unitario, relativo a la tecnología, y da soporte al equipo. Este cuadrante contiene pruebas unitarias. Estas pruebas deben automatizarse e incluirse en el sistema de integración continua.
- *El cuadrante Q2* es el nivel de sistema, orientadas al negocio y confirma el comportamiento del producto. Este cuadrante contiene pruebas funcionales, ejemplos, pruebas de historias, prototipos de experiencia del usuario y simulaciones. Estas pruebas comprueban los criterios de aceptación y pueden ser manuales o estar automatizadas. A menudo se crean durante el desarrollo de historias de usuario y por lo tanto mejoran la calidad de las historias. Son útiles a la hora de crear juegos de pruebas de regresión automatizadas.
- *El cuadrante Q3* es el nivel de sistema o aceptación de usuario, relativo al negocio, y contiene pruebas que evalúan el producto mediante escenarios y datos realistas. Este cuadrante incluye pruebas exploratorias, escenarios, flujos de proceso, pruebas de usabilidad, pruebas de aceptación de usuario, pruebas alfa y pruebas beta. A menudo estas pruebas son manuales y están orientadas hacia el usuario.
- *El cuadrante Q4* es el nivel de sistema o aceptación operativa, relativo a la tecnología, y contiene pruebas que evalúan el producto. Este cuadrante contiene pruebas de rendimiento, carga, estrés y escalabilidad, pruebas de seguridad, pruebas de mantenibilidad, gestión de la memoria, compatibilidad e interoperabilidad, migración de datos, infraestructura y recuperación. A menudo están automatizadas.

Durante una iteración dada, es posible que se requieran pruebas de cualquiera o de todos los cuadrantes. Los cuadrantes de pruebas son aplicables a pruebas dinámicas más que a pruebas estáticas.



1.4 Funciones de un probador

A lo largo de este plan de estudios, nos hemos referido a los métodos y técnicas ágiles, y el papel de un probador dentro de los distintos ciclos de vida ágiles. Esta subsección está dedicada específicamente al papel de un probador en un proyecto siguiendo un ciclo de vida Scrum [Aalst13].

Trabajo en equipo

El trabajo en equipo es un principio fundamental del desarrollo ágil. Los procesos ágiles enfatizan el enfoque de equipo completo, compuesto por desarrolladores, probadores y representantes de negocio trabajando juntos. A continuación se indican buenas prácticas organizativas y de comportamiento en equipos Scrum:

- *Multidisciplinar*: Cada miembro del equipo aporta un conjunto de habilidades distintas al equipo. Los miembros del equipo trabajan juntos sobre la estrategia de pruebas, la planificación de pruebas, la especificación de pruebas, la ejecución de pruebas, la evaluación de pruebas y la comunicación de los resultados de pruebas.
- *Auto-organizado:* El equipo puede constar exclusivamente de desarrolladores, pero, según se indica en la Sección 2.1.5, idealmente debe haber uno o más probadores.
- *Co-ubicado:* Los probadores se sientan con los desarrolladores y con el propietario del producto.
- Colaborativo: Los probadores colaboran con los miembros de su equipo, con otros equipos, con las partes interesadas, con el propietario del producto y con el Scrum Master.
- *Empowered:* El equipo adopta globalmente decisiones técnicas sobre el diseño y las pruebas (desarrolladores, probadores y Scrum Master), en colaboración con el propietario del producto y otros equipos, si procede.
- *Comprometido:* El probador está comprometido a cuestionar y evaluar el comportamiento y las características del producto por lo que respecta a las expectativas de los clientes y usuarios.
- *Transparente:* El progreso del desarrollo y de las pruebas es visible en el tablero de tareas ágil (remítase a la Sección 2.2.1).
- *Creíble:* El probador debe asegurar la credibilidad de la estrategia elegida para las pruebas, su implementación y ejecución, de lo contrario las partes interesadas no confiarán en los resultados de las pruebas. A menudo esto se realiza facilitando información a las partes interesadas sobre el proceso de prueba.
- *Abierto al feedback:* El feedback es un aspecto importante del éxito en cualquier proyecto, especialmente en los proyectos ágiles. Las reuniones retrospectivas permiten a los equipos aprender de los éxitos y de los fracasos.
- *Flexible:* Las pruebas deben ser capaces de responder ante los cambios, al igual que el resto de actividades en los proyectos ágiles.

Estas buenas prácticas maximizan la probabilidad de éxito en las pruebas en proyectos Scrum

El sprint cero

El sprint cero es la primera iteración del proyecto en la que se llevan a cabo muchas actividades de preparación (remítase a la Sección 1.2.5). El probador colabora con el equipo en las siguientes actividades durante esta iteración:

- Identificar el alcance del proyecto (es decir, el backlog del producto)
- Crear una arquitectura de sistema inicial y prototipos de alto nivel
- Planificar, obtener e instalar las herramientas necesarias (por ejemplo, para la gestión de pruebas, la gestión de defectos, la automatización de las pruebas y la integración continua)
- Crear una estrategia de pruebas inicial para todos los niveles de las pruebas, que aborde (entre otros temas), el alcance de las pruebas, los riesgos técnicos, los tipos de pruebas (remítase a la Sección 3.1.3) y los objetivos de cobertura
- Llevar a cabo un análisis inicial de riesgos de calidad (remítase a la Sección 3.2.1)
- Definir las métricas de las pruebas para medir el proceso de prueba, el progreso de las pruebas en el proyecto y la calidad del producto
- Especificar la definición de "hecho"
- Crear el tablero de tareas (véase la Sección 2.2.1)
- Definir cuándo continuar o cuando detener las pruebas antes de entregar el sistema al cliente

El sprint cero establece la dirección de lo que tienen que conseguir las pruebas y cómo tienen que conseguirlo a lo largo de los sprints.

Integración

En los proyectos ágiles, el objetivo es entregar valor al cliente de una forma continua (preferiblemente en todos los sprints). Para ello, la estrategia de integración debe tener en cuenta tanto el diseño como las pruebas. Para poder implementar una estrategia de pruebas continua en las funcionalidades y características entregadas, es importante definir todas las dependencias entre las funciones y funcionalidades subyacentes.

Planificación de las pruebas

Ya que las pruebas están totalmente integradas en el equipo ágil, la planificación de las pruebas debe empezar durante la sesión de planificación de la entrega y actualizarse durante cada sprint. La planificación de las pruebas para la entrega y para cada sprint debe abordar los aspectos referidos en la Sección 1.2.5.

La planificación del sprint da lugar a un conjunto de tareas a incluir en el tablero de tareas, donde cada tarea debe tener una duración de uno o dos días de trabajo. Además, debe hacerse un seguimiento de todos los problemas referidos a las pruebas para mantener un flujo de pruebas constante.

Prácticas de pruebas ágiles

Muchas prácticas pueden ser de utilidad para los probadores en un equipo scrum, algunas de las cuales incluyen:

- Por parejas: Dos miembros del equipo (por ejemplo, un probador y un desarrollador, dos probadores o un probador y un propietario del producto) se sientan juntos para llevar a cabo pruebas u otra tarea del sprint.
- Diseño de pruebas incrementales: Los casos de pruebas se construyen gradualmente a partir de historias de usuario y demás bases de pruebas, empezando con pruebas sencillas y pasando a pruebas más complejas.
- Mapas mentales: Los mapas mentales son una herramienta útil a la hora de hacer pruebas [Crispin08]. Por ejemplo, los probadores pueden usar mapas mentales para identificar qué sesiones de prueba llevar a cabo, para mostrar las estrategias de pruebas y para describir los datos de prueba.

Estas prácticas son adicionales a otras prácticas descritas en este Plan de estudios y en el Capítulo 4 del Plan de estudios de Nivel Básico [ISTQB_FL_SYL].

2. Evaluar riesgos de calidad y estimar el esfuerzo de prueba

Un objetivo típico de las pruebas en todos los proyectos, ágiles o tradicionales, es reducir el riesgo de los problemas de calidad del producto a un nivel aceptable antes de la entrega. Los probadores de proyectos ágiles pueden utilizar el mismo tipo de técnicas que se utilizan en los proyectos tradicionales para identificar los riesgos de calidad (o riesgos de producto), evaluar el nivel asociado de riesgo, estimar el esfuerzo requerido para reducir esos riesgos de manera suficiente y a continuación mitigar esos riesgos a través del diseño, la implementación y la ejecución de pruebas. No obstante, a la vista de la brevedad de las iteraciones y del ritmo de los cambios en proyectos ágiles, se requieren algunas adaptaciones de esas técnicas.

2.1 Evaluar los riesgos de calidad en proyectos ágiles

Uno de los muchos retos de las pruebas es la correcta selección, asignación y priorización de las condiciones de prueba. Esto incluye determinar la cantidad de esfuerzo que debe dedicarse para cubrir cada condición con pruebas, y secuenciar las pruebas resultantes de tal manera que se optimice la eficacia y eficiencia del trabajo de pruebas a realizar. Los probadores en equipos ágiles pueden recurrir a estrategias de identificación de riesgos, análisis de riesgos y mitigación de riesgos para ayudar a establecer un número aceptable de

casos de prueba a ejecutar, si bien pueden darse muchas restricciones y variables que pueden requerir compromisos.

El riesgo es la posibilidad de un resultado o evento, negativo o no deseado. El nivel de riesgo se establece evaluando la probabilidad de que el riesgo suceda y su impacto. Cuando el efecto principal del problema potencial se refiere a la calidad del producto, los problemas potenciales se denominan riesgos de calidad o riesgos de producto. Cuando el efecto principal del problema potencial se refiere al éxito del proyecto, los problemas potenciales se denominan riesgos de proyecto o riesgos de planificación. [Black07] [vanVeenendaal12].

En los proyectos ágiles, el análisis de riesgos de calidad se lleva a cabo en dos momentos.

- Planificación de la entrega: los representantes de negocio que conocen las funcionalidades en la entrega proporcionan una perspectiva de alto nivel de los riesgos, y todo el equipo, incluidos los probadores, pueden ayudar a identificar y evaluar los riesgos.
- Planificación de la iteración: todo el equipo identifica y evalúa los riesgos de calidad.

Algunos ejemplos de los riesgos de calidad de un sistema incluyen:

- Cálculos incorrectos en informes (un riesgo funcional asociado a la exactitud)
- Respuesta lenta a entradas del usuario (un riesgo no funcional asociado a la eficiencia y al tiempo de respuesta)
- Dificultad para comprender pantallas y campos (un riesgo no funcional asociado a la usabilidad y a la facilidad de comprensión).

Según se ha mencionado antes, una iteración empieza con la planificación de la iteración, que culmina en tareas estimadas en un tablero de tareas. Estas tareas pueden priorizarse en parte en base al nivel de los riesgos de calidad asociados a las mismas. Las tareas asociadas a riesgos más altos deben empezar antes y suponer un esfuerzo de prueba mayor. Las tareas asociadas a riesgos más bajos deben empezar más tarde y suponer un esfuerzo de prueba menor.

A continuación se muestra un ejemplo de cómo en un proyecto ágil puede llevarse a cabo el proceso de análisis de los riesgos de calidad durante la planificación de la iteración

1. Reunir a los miembros del equipo, incluidos los probadores.
2. Enumerar todos los elementos del backlog para la iteración actual (por ejemplo, en un tablero de tareas).
3. Identificar los riesgos de calidad asociados a cada elemento, teniendo en cuenta todas las características de calidad relevantes.
4. Evaluar cada riesgo identificado, lo que incluye dos actividades: categorizar el riesgo y determinar su nivel de riesgo en base al impacto y a la probabilidad de los defectos.
5. Determinar el alcance de las pruebas de manera proporcional al nivel de riesgo.

6. Seleccionar la técnica o técnicas de pruebas adecuadas para mitigar cada riesgo, en función del nivel de riesgo y de la característica de calidad relevante.

A continuación, el probador diseña, implementa y ejecuta pruebas para mitigar los riesgos. Esto incluye todas las funcionalidades, comportamientos, características de calidad y atributos que afectan a la satisfacción del cliente, del usuario y de las partes interesadas.

A lo largo del proyecto, el equipo debe conocer cualquier información adicional que pueda alterar el conjunto de riesgos y/o el nivel de riesgo asociado a riesgos de calidad conocidos. Debe realizarse un ajuste periódico del análisis de riesgos de calidad, y los consiguientes ajustes a las pruebas. Los ajustes incluyen identificar nuevos riesgos, volver a analizar el nivel de los riesgos existentes y evaluar la eficacia de las actividades de mitigación de riesgos.

Los riesgos de calidad también pueden mitigarse antes de empezar a ejecutar las pruebas. Por ejemplo, si se detectan problemas con las historias de usuario durante la identificación de riesgos, el equipo del proyecto puede llevar a cabo revisiones exhaustivas de las historias de usuario como estrategia de mitigación.

2.2 Estimación del esfuerzo de prueba en base al contenido y al riesgo

Durante la planificación de la entrega, el equipo ágil estima el esfuerzo necesario para completar la entrega. Esta estimación aborda también el esfuerzo de las pruebas. Una técnica de estimación habitual que se utiliza en los proyectos ágiles es la planificación de póquer, una técnica basada en el consenso. El propietario del producto o el cliente lee una historia de usuario al equipo. Cada componente del equipo tiene una baraja de cartas con valores similares a la secuencia Fibonacci (es decir, 0, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, …) o cualquier otra progresión elegida (por ejemplo, tamaños de camisetas que van de XS a XL). El valor representa el número de puntos de historia, los días de esfuerzo o cualquier otra unidad con la que el equipo realice los cálculos. Se recomienda la secuencia Fibonacci porque los números en la secuencia reflejan que la incertidumbre aumenta de manera proporcional al tamaño de la historia. Una estimación alta normalmente significa que la historia no se ha entendido bien o debería dividirse en varias historias más pequeñas.

Los miembros del equipo discuten la funcionalidad, y plantean preguntas al propietario del producto en caso necesario. En la estimación intervienen aspectos como el esfuerzo de desarrollo y el esfuerzo de las pruebas, la complejidad de la historia y el alcance de las pruebas. Por lo tanto, es recomendable incluir el nivel de riesgo de un elemento del backlog, además de la prioridad especificada por el propietario del producto, antes de iniciar la sesión de planificación de póquer. Tras debatir en profundidad sobre la funcionalidad, cada miembro del equipo selecciona en privado una carta que represente su estimación. A continuación, se enseñan todas las cartas a la vez. Si todos los componentes del equipo han seleccionado el mismo valor, entonces esa será la estimación. De lo contrario, el equipo

tratará las diferencias en las estimaciones y se repetirá la estimación de póquer hasta que se llegue a un acuerdo, bien por consenso, bien mediante la aplicación de reglas (por ejemplo, utilizar la mediana, utilizar la puntuación más alta) para limitar el número de rondas de la planificación de póquer. Estos debates aseguran una estimación fiable del esfuerzo necesario para completar los elementos del backlog del producto requeridos por el propietario del producto y ayudan a mejorar el conocimiento colectivo de qué tiene que hacerse [Cohn04].

3. Técnicas en los proyectos ágiles

Muchas de las técnicas de pruebas y niveles de pruebas que son aplicables a los proyectos tradicionales también pueden aplicarse a los proyectos ágiles. Sin embargo, en los proyectos ágiles, deben tenerse en cuenta ciertas consideraciones específicas y diferencias en cuanto a técnicas, terminologías y documentación de pruebas.

3.1 Criterios de aceptación, cobertura adecuada y otra información para pruebas

Los proyectos ágiles perfilan los requisitos iniciales como historias de usuario en un backlog priorizado al inicio del proyecto. Los requisitos iniciales son breves y normalmente siguen un formato predefinido (remítase a la Sección 1.2.2). Los requisitos no funcionales, como usabilidad y rendimiento, también son importantes y pueden especificarse como historias de usuario únicas o relacionadas con otras historias de usuario funcionales. Los requisitos no funcionales pueden seguir un formato predefinido o normalizado, como el de [ISO25000], o estándares específicos de la industria.

Las historias de usuario son una parte importante de la base de pruebas. Otras posibles bases de pruebas son:

- Experiencia de proyectos anteriores
- Funciones, funcionalidades y características de calidad del sistema existentes
- Código, arquitectura y diseño
- Perfiles de usuario (contexto, configuraciones del sistema y comportamiento del usuario)
- Información sobre defectos procedentes de proyectos existentes y anteriores
- Una categorización de los defectos en una taxonomía de defectos
- Normativas aplicables (por ejemplo, [DO-178B] para software de aviónica)
- Riesgos de calidad (remítase a la Sección 3.2.1)

Durante cada iteración, los desarrolladores crean código que implementa las funciones y funcionalidades descritas en las historias de usuario, con las características de calidad relevantes y este código se verifica y valida a través de pruebas de aceptación. Para ser

testeables, los criterios de aceptación deben abordar los siguientes temas cuando proceda [Wiegers13]:

- Comportamiento funcional: El comportamiento debe ser observable externamente con acciones de usuario que operan bajo ciertas configuraciones.
- Características de calidad: Cómo el sistema realiza el comportamiento especificado. Las características también pueden denominarse atributos de calidad o requisitos no funcionales. Algunas características de calidad habituales son rendimiento, fiabilidad, usabilidad, etc.
- Escenarios (casos de uso): Una secuencia de acciones entre un actor externo (a menudo un usuario) y el sistema, para conseguir un objetivo específico o una tarea de negocio.
- Normas de negocio: Las actividades que solo pueden realizarse en el sistema bajo ciertas condiciones definidas por procedimientos y restricciones externos (por ejemplo, los procedimientos utilizados por una compañía aseguradora para gestionar partes de seguros).
- Interfaces externas: Descripciones de las conexiones entre el sistema a desarrollar y el mundo exterior. Las interfaces externas pueden dividirse en varios tipos (interfaz de usuario, interfaz con otros sistemas, etc.).
- Restricciones: Cualquier restricción de diseño e implementación que pueda restringir las opciones del desarrollador. Los dispositivos con software empotrado a menudo deben cumplir restricciones físicas tales como el tamaño, el peso y las conexiones de interfaz.
- Definiciones de datos: El cliente puede describir el formato, el tipo de datos, los valores permitidos y los valores por defecto para un elemento de datos en la composición de una estructura compleja de datos de negocio.

Además de las historias de usuario y sus criterios de aceptación asociados, hay más información relevante para el probador, entre la que encontramos:

- Cómo se supone que el sistema va a funcionar y utilizarse
- Las interfaces de sistema que pueden utilizarse o están accesibles para probar el sistema
- Si la herramienta de soporte actual es suficiente
- Si el probador dispone del conocimiento y las habilidades suficientes para llevar a cabo las pruebas necesarias

Los probadores a menudo descubrirán la necesidad de información adicional (por ejemplo, cobertura de código) a lo largo de las iteraciones y deberán trabajar de manera colaborativa con el resto de los miembros del equipo ágil para obtener dicha información. La información relevante es importante para determinar si una actividad específica puede considerarse hecha. Este concepto de la definición de "hecho" es crítico en los proyectos ágiles y se aplica de varias formas distintas según se verá en las siguientes subsecciones.

Niveles de pruebas

Cada nivel de pruebas tiene su propia definición de "hecho". La siguiente lista incluye ejemplos que pueden ser relevantes para los distintos niveles de pruebas.

Pruebas unitarias

- 100% de cobertura de decisión cuando sea posible, con revisiones minuciosas de las rutas inviables.
- Se ha efectuado un análisis estático en todo el código
- No hay ningún defecto importante sin resolver (ordenado por prioridad y severidad)
- No se conoce deuda técnica inaceptable y conocida en el diseño y el código [Jones11]
- Todo el código, todas las pruebas unitarias y todos los resultados de las pruebas unitarias se han revisado
- Todas las pruebas unitarias están automatizadas
- Las características importantes están dentro de los límites acordados (por ejemplo, rendimiento)

Pruebas de integración

- Se han probado todos los requisitos funcionales, incluyendo pruebas tanto positivas como negativas, con el número de pruebas en función del tamaño, la complejidad y los riesgos
- Se han probado todas las interfaces entre unidades
- Se han cubierto todos los riesgos de calidad en conformidad con el alcance acordado de las pruebas
- No hay defectos importantes sin resolver (priorizados por riesgo e importancia)
- Se han reportado todos los defectos encontrados
- Se han automatizado todas las pruebas de regresión, cuando sea posible, y todas las pruebas automatizadas están almacenadas en un repositorio común

Pruebas de sistema

- Se han realizado pruebas de extremo a extremo de las historias de usuario, funcionalidades y funciones
- Se han cubierto todos los modelos de usuarios
- Se han cubierto las características de calidad del sistema más importantes (por ejemplo, rendimiento, robustez, fiabilidad)
- Se han hecho pruebas en un entorno parecido al de producción, incluyendo todo el hardware y el software para todas las configuraciones soportadas, en la medida de los posible
- Se han cubierto todos los riesgos de calidad de conformidad con el alcance acordado de las pruebas

- Se han automatizado todas las pruebas de regresión, cuando sea posible, y todas las pruebas automatizadas están almacenadas en un repositorio común
- Se han reportado y posiblemente arreglado todos los defectos encontrados
- No hay defectos importantes sin resolver (priorizados por riesgo e importancia)

Historia de usuario
La definición de "hecho" para las historias de usuario puede venir determinada por los siguientes criterios:

- Las historias de usuario seleccionadas para la iteración están completas, el equipo las ha entendido y cuentan con criterios de aceptación detallados y testeables
- Todos los elementos de la historia de usuario se han especificado y revisado, incluyendo las pruebas de aceptación de la historia de usuario
- El equipo ha identificado y estimado las tareas necesarias para implementar y probar las historias de usuario seleccionadas

Prestación
La definición de "hecho" para las prestaciones, pueden abarcar varias historias de usuario o épicas y puede incluir:

- El cliente ha definido y aprobado todas las historias de usuario que integran la funcionalidad, con criterios de aceptación.
- El diseño está completo, sin deuda técnica conocida
- El código está completo, sin deuda técnica conocida ni refactorización inacabada
- Se han realizado pruebas unitarias y se ha logrado el nivel definido de cobertura
- Se han realizado las pruebas de integración y las pruebas de sistema para la funcionalidad según los criterios de cobertura definidos
- No hay defectos importantes pendientes de corregir
- La documentación de la funcionalidad está completa, lo que puede incluir notas de la entrega, manuales de usuario y funciones de ayuda on-line

Iteración
La definición de "hecho" para la iteración puede incluir los siguientes puntos:

- Todas las funcionalidades de la iteración están listas y han sido probadas individualmente de conformidad con los criterios de nivel de la funcionalidad
- Todos los defectos no críticos que no pueden arreglarse dentro de las restricciones de la iteración se han añadido al backlog del producto y se han priorizado
- Se ha completado y probado la integración de todas las funcionalidades para la iteración
- Se ha escrito, revisado y aprobado la documentación

En este punto, el software es potencialmente entregable porque la iteración se ha completado con éxito, pero no todas las iteraciones tienen como resultado una entrega.

Entrega

La definición de "hecho" para una entrega, que puede abarcar varias iteraciones, puede incluir las siguientes áreas:

- Cobertura: Se han cubierto mediante pruebas todos los elementos de la base de prueba relevantes para todos los contenidos de la entrega. La idoneidad de la cobertura viene determinada por lo nuevo o modificado, su complejidad y sus dimensiones, así como los riesgos de fallo asociados.
- Calidad: La intensidad de los defectos (por ejemplo, cuántos defectos se encuentran por día o por transacción), la densidad de los defectos (por ejemplo, el número de defectos encontrados en comparación con el número de historias de usuario, el esfuerzo y/o los atributos de calidad) y el número estimado de defectos pendientes están dentro de unos límites aceptables, se han entendido las consecuencias de los defectos no resueltos y pendientes (por ejemplo, la severidad y la prioridad) y son aceptables, se ha entendido el nivel de riesgo residual asociado a cada riesgo de calidad identificado y es aceptable.
- Tiempo: Si se ha alcanzado la fecha de entrega prefijada, deben tenerse en cuenta las consideraciones de negocio asociadas a la entrega.
- Coste: El coste del ciclo de vida estimado debe utilizarse para calcular el retorno de la inversión para el sistema entregado (es decir, el desarrollo calculado y el coste de mantenimiento debe ser significativamente menor que las ventas totales esperadas del producto). La parte más importante del coste de ciclo de vida a menudo se refiere a tareas de mantenimiento una vez entregado el producto, debido al número de defectos que han pasado a producción.

3.2 Aplicación del desarrollo guiado por pruebas de aceptación

El desarrollo guiado por pruebas de aceptación es un enfoque de "probar primero". Los casos de prueba se crean antes de implementar la historia de usuario. Los casos de prueba los crea el equipo ágil, incluyendo el desarrollador, el probador y los representantes de negocio [Adzic09] y pueden ser manuales o automatizados. El primer paso es un taller de especificación en el que se analiza, se debate y se escribe la historia de usuario por parte de desarrolladores, probadores y representantes de negocio. Todas las inconclusiones, ambigüedades o errores en la historia de usuario se arreglan durante este proceso.

El siguiente paso es crear las pruebas. Esto puede hacerlo el equipo en conjunto o el probador a nivel individual. 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. Estos ejemplos ayudarán al equipo a implementar la historia de usuario correctamente. Dado que los ejemplos y las pruebas coinciden, a menudo estos términos se utilizan de manera intercambiable. El trabajo empieza con ejemplos básicos y preguntas abiertas.

Normalmente, las primeras pruebas son las pruebas positivas, que confirman el comportamiento correcto sin condiciones de error, incluyendo la secuencia de actividades ejecutada si todo va según lo previsto. Una vez realizadas las pruebas positivas, el equipo debe escribir pruebas negativas y cubrir atributos no funcionales también (por ejemplo, rendimiento, usabilidad). Las pruebas se expresan de forma que todas las partes interesadas pueden entender, con sentencias en lenguaje natural que conllevan las precondiciones necesarias, si existen, las entradas y las salidas asociadas.

Los ejemplos deben cubrir todas las características de la historia de usuario pero ninguna característica que no exista en la historia. Esto significa que no debe existir un ejemplo que describa un aspecto de la historia de usuario que no esté documentado en la propia historia. Además, no debe haber dos ejemplos que describan las mismas características de la historia de usuario.

3.3 Diseño de pruebas de caja negra funcionales y no funcionales

En las pruebas ágiles, los probadores crean las pruebas en paralelo a las actividades de programación de los desarrolladores. Al igual que los desarrolladores programan en base a las historias de usuario y a los criterios de aceptación, los probadores crean pruebas en base a las historias de usuario y a sus criterios de aceptación. (Algunas pruebas, como las pruebas exploratorias y otras pruebas basadas en la experiencia, se crean más tarde, durante la ejecución de las pruebas, según se explica en la Sección 3.3.4). Los probadores pueden aplicar técnicas de diseño de pruebas de caja negra tradicionales como la partición de equivalencias, el análisis de valores límite, las tablas de decisión y las pruebas de transición de estado para crear estas pruebas. Por ejemplo, se podría utilizar el análisis de valores límite para seleccionar los valores de prueba cuando un cliente está limitado en cuanto al número de elementos que pueden seleccionar para comprar.

En muchas situaciones, los requisitos no funcionales pueden documentarse como historias de usuario. Las técnicas de diseño de pruebas de caja negra (como el análisis de valores límite) también pueden utilizarse para crear pruebas para características de calidad no funcionales. La historia de usuario puede contener requisitos de rendimiento o fiabilidad. Por ejemplo, una ejecución dada no puede exceder un límite de tiempo o un número de operaciones puede fallar menos de un cierto número de veces.

Para más información sobre el uso de las técnicas de diseño de pruebas de caja negra, remítase al Plan de estudios de Nivel Básico [ISTQB_FL_SYL] y al Plan de estudios de Analista de Pruebas de Nivel Avanzado [ISTQB_ALTA_SYL].

3.4 Pruebas exploratorias y pruebas ágiles

Las pruebas exploratorias son importantes en los proyectos ágiles debido al tiempo limitado disponible para el análisis de pruebas y los detalles limitados de las historias de usuario. Para lograr los mejores resultados, las pruebas exploratorias deben combinarse con otras técnicas basadas en la experiencia como parte de una estrategia de pruebas reactiva, en combinación con otras estrategias de pruebas como las pruebas basadas en riesgos, pruebas basadas en requisitos analíticos y pruebas basadas en modelos. Las estrategias de pruebas y la combinación de estrategias de pruebas se abordan en el Plan de estudios de Nivel Básico [ISTQB_FL_SYL].

En las pruebas exploratorias, el diseño de pruebas y la ejecución de pruebas tienen lugar a la vez, guiadas por un contrato de pruebas elaborado. Un contrato de pruebas establece las condiciones de prueba a cubrir durante sesiones de pruebas limitadas en el tiempo. Durante las pruebas exploratorias, los resultados de las últimas pruebas guían la siguiente prueba. Pueden utilizarse las mismas técnicas de caja negra y caja blanca para diseñar las pruebas que las utilizadas para las pruebas prediseñadas.
Un contrato de pruebas puede incluir la siguiente información:

- Actor: usuario previsto del sistema
- Propósito: el tema del contrato, incluyendo qué objetivo particular quiere alcanzar el actor, es decir, las condiciones de prueba
- Preparación: qué tiene que haber para poder iniciar la ejecución de la prueba
- Prioridad: importancia relativa de este contrato, en base a la prioridad de la historia de usuario asociada o del nivel de riesgo
- Referencias: especificaciones (por ejemplo, historia de usuario), riesgos, u otras fuentes de información
- Datos: los datos necesarios para llevar a cabo el contrato
- Actividades: una lista de ideas de lo que el actor puede querer hacer con el sistema (por ejemplo, "Acceder al sistema como superusuario") y qué sería interesante probar (pruebas tanto positivas como negativas)
- Notas de oráculo: cómo evaluar el producto para establecer los resultados correctos (por ejemplo, capturar qué pasa en la pantalla y compararlo con lo que está escrito en el manual de usuario).
- Variaciones: acciones y evaluaciones alternativas para complementar las ideas descritas bajo actividades

Para gestionar las pruebas exploratorias, puede utilizarse un método denominado gestión de pruebas basada en sesiones. Por sesión se entiende un periodo ininterrumpido de pruebas que podría durar de 60 a 120 minutos. Las sesiones de pruebas incluyen lo siguiente:

- Sesión de reconocimiento (para aprender cómo funciona)
- Sesión de análisis (evaluación de la funcionalidad o características)
- Cobertura exhaustiva (casos extremos, escenarios, interacciones)

La calidad de las pruebas depende de la habilidad de los probadores para hacer las preguntas pertinentes sobre qué probar. Se pueden mencionar los siguientes ejemplos:

- ¿Qué es lo más importante a averiguar del sistema?
- ¿De qué forma puede fallar el sistema?
- ¿Qué sucede si...?
- ¿Qué debería suceder cuando...?
- ¿Se cumplen las necesidades, requisitos y expectativas del cliente?
- ¿Se puede instalar el sistema (y retirar si procede) en todas las rutas de actualización soportadas?

Durante la ejecución de las pruebas, el probador aporta creatividad, intuición y pericia para detectar posibles problemas en el producto. El probador también debe tener un buen conocimiento y comprensión del software sujeto a pruebas, el ámbito de negocio, cómo se utiliza el software y cómo determinar cuándo falla el sistema.

Durante las pruebas puede aplicarse una serie de heurísticas. Una heurística puede guiar al probador sobre cómo llevar a cabo las pruebas y evaluar los resultados [Hendrickson]. Entre otros ejemplos, destacan:

- Valores Límites
- CRUD (Create, Read, Update, Delete: Crear, Leer, Actualizar, Borrar)
- Variaciones en la configuración
- Interrupciones (por ejemplo, desconexión, caída o reinicio)

Es importante que el probador documente el proceso en la máxima medida posible. De lo contrario, será difícil volver y ver cómo se ha descubierto un problema en el sistema. La siguiente lista ofrece ejemplos de información que puede ser útil documentar:

- Cobertura de pruebas: qué entrada se ha utilizado, cuánto se ha cubierto y cuánto queda pendiente de probar
- Notas de evaluación: observaciones durante las pruebas, ¿el sistema y la funcionalidad objeto de la prueba parecen estables?, ¿se ha encontrado algún defecto?, ¿qué se prevé como pasos siguientes según las observaciones actuales?, y cualquier otra lista de ideas.
- Lista de riesgos/estrategia: ¿Qué riesgos se han cubierto y qué riesgos siguen pendientes de cubrir entre los más importantes?, ¿se seguirá la estrategia inicial?, ¿necesita algún cambio?
- Problemas, preguntas y anomalías: cualquier comportamiento imprevisto, cualquier pregunta sobre la eficiencia del enfoque, cualquier inquietud sobre las ideas/intentos de la prueba, el entorno de prueba, los datos de prueba, mala interpretación de la función, de script de prueba o del sistema bajo prueba.
- Comportamiento real: registro de comportamiento real del sistema que debe guardarse (por ejemplo, vídeo, impresiones de pantalla, ficheros de datos de salida)

La información registrada debe capturarse y/o resumirse mediante algún tipo de herramienta de gestión de estados (por ejemplo, herramientas de gestión de pruebas, herramienta de gestión de tareas, el tablero de tareas), de forma que resulte fácil para las partes interesadas conocer el estado actual de todas las pruebas realizadas.

4. Herramientas en proyectos ágiles

Las herramientas descritas en el plan de estudios de Nivel Básico [ISTQB_FL_SYL] son relevantes y las utilizan los probadores en equipos ágiles. No todas las herramientas se utilizan de la misma forma y algunas herramientas tienen más relevancia para los proyectos ágiles que en los proyectos tradicionales. Por ejemplo, si bien las herramienta de gestión de pruebas, las herramientas de gestión de requisitos, y las herramientas de gestión de incidencias (herramientas de seguimiento de defectos) pueden utilizarse en equipos ágiles, algunos equipos ágiles optan por herramientas integrales (por ejemplo, gestión del ciclo de vida de las aplicaciones o gestión de tareas) que ofrecen prestaciones pertinentes para el desarrollo ágil, como los tableros de tareas, los gráficos Burndown, y las historias de usuario. Las herramientas de gestión de la configuración son importantes para los probadores de equipos ágiles dada la gran cantidad de pruebas automatizadas en todos los niveles y la necesidad de almacenar y gestionar los artefactos de prueba automatizados asociados.

Además de las herramientas descritas en el Plan de estudios de Nivel Básico [ISTQB_FL_SYL], los probadores en proyectos ágiles también pueden utilizar las herramientas que se describen en las siguientes subsecciones. Estas herramientas las utilizan todo el equipo para asegurar la colaboración y la puesta en común de información en el equipo, que son prácticas clave de los procesos ágiles.

4.1 Herramientas de gestión y seguimiento de tareas

En algunos casos, los equipos ágiles utilizan tableros físicos de historias /tareas (por ejemplo, pizarra, tablero de corcho) para gestionar y hacer un seguimiento de las historias de usuario, las pruebas y demás tareas a lo largo de cada sprint. Otros equipos utilizan software de gestión del ciclo de vida de las aplicaciones y gestión de tareas, incluyendo tableros de tareas electrónicos. Estas herramientas sirven para lo siguiente:

- Grabar historias y sus tareas relevantes de desarrollo y tareas de pruebas, para garantizar que no se pierde nada durante un sprint.
- Recoger las estimaciones de los miembros del equipo sobre sus tareas y calcular automáticamente el esfuerzo necesario para implementar una historia y así dar soporte para conseguir sesiones de planificación de iteraciones más eficientes
- Asociar tareas de desarrollo y tareas de prueba a la misma historia, para ofrecer una imagen completa del esfuerzo requerido del equipo para implementar la historia

- Incorporar actualizaciones del desarrollador y del probador al estado de la tarea a medida que van completando su trabajo, ofreciendo automáticamente una instantánea calculada del estado de cada historia, la iteración y la entrega en general
- Ofrecer una representación visual (a través de métricas, gráficos y cuadros de mando de pruebas) del estado actual de cada historia de usuario, la iteración y la entrega, permitiendo a todas las partes interesadas, incluyendo a personas en equipos deslocalizados geográficamente, comprobar el estado fácilmente
- Integrarse con las herramientas de gestión de la configuración, para permitir el registro automatizado de los check-in de código y builds contra tareas, y, en algunos casos, la actualización del estado de las tareas

4.2 Herramientas de comunicación y para compartir información

Además del correo electrónico, los documentos y la comunicación verbal, los equipos ágiles a menudo utilizan tres tipos de herramientas para fomentar y compartir la información: wikis, mensajería instantánea y compartición de escritorios.

Las wikis permiten a los equipos construir y compartir una base de conocimiento on-line en base a varios aspectos del proyecto, incluidos los siguientes:

- Diagramas, grupos de debate, imágenes relacionadas con características relevantes, prototipos u otra información
- Herramientas y/o técnicas de desarrollo y pruebas que sean útiles para otros miembros del equipo
- Métricas, gráficos y cuadros de mando sobre el estado del producto, que son especialmente útiles cuando la wiki está integrada en otras herramientas como un servidor de compilación y el sistema de gestión de tareas, ya que la herramienta puede actualizar automáticamente el estado del producto
- Conversaciones entre miembros del equipo parecidas a la mensajería instantánea y al correo electrónico, compartiéndolo de esta forma con todos los miembros del equipo

Los sistemas de mensajería instantánea, audio conferencia y herramientas de chat con vídeo ofrecen las siguientes ventajas:

- Permiten una comunicación directa en tiempo real entre los miembros del equipo, especialmente en el caso de equipos deslocalizados
- Implican a los equipos deslocalizados en las reuniones de seguimiento
- Reducen la factura telefónica al utilizar tecnología de voz sobre IP, eliminando las restricciones de coste que podrían limitar la comunicación entre los miembros del equipo en ubicaciones distribuidas

Las herramientas de compartición y captura de escritorios ofrecen las siguientes ventajas:

- En los equipos deslocalizados pueden realizarse demostraciones de producto, revisiones de código e incluso trabajo por pares

- Capturar demostraciones de producto al término de cada iteración, lo que puede 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.

4.3 Herramientas de construcción y distribución de software

Según lo descrito anteriormente en este plan de estudios, la construcción diaria y el despliegue del software constituyen una práctica clave en los equipos ágiles. Para ello es necesario el uso de herramientas de integración continua y herramientas de distribución de versiones. Los usos, ventajas y riesgos de estas herramientas se han descrito en la Sección 1.2.4.

4.4 Herramientas de gestión de la configuración

En los equipos ágiles, se pueden utilizar herramientas de gestión de la configuración no solo para almacenar código fuente y pruebas automatizadas, sino que a menudo también se almacenan pruebas manuales y otros productos de trabajo de pruebas en el mismo repositorio que el código fuente del producto. Esto proporciona trazabilidad para saber entre qué versiones del software se han probado y con qué versiones específicas de las pruebas, y permite el cambio rápido sin perder información histórica. Los principales tipos de sistemas de control de versiones incluyen sistemas centralizados de control de código fuente y sistemas deslocalizados de control de versiones. Las dimensiones del equipo, su estructura, ubicación y requisitos para integrarse con otras herramientas determinarán qué sistema de control de versiones es el adecuado para un proyecto ágil en particular.

4.5 Herramientas de diseño, implementación y ejecución de pruebas

Algunas herramientas son de utilidad para los probadores ágiles en momentos específicos del proceso de pruebas de software. Si bien la mayor parte de estas herramientas no son nuevas ni específicas de los entornos ágiles, ofrecen capacidades importantes a la vista de la rapidez de los cambios en los proyectos ágiles.
Herramientas de diseño de pruebas: El uso de herramientas como los mapas mentales han ganado popularidad para diseñar y definir pruebas rápidamente para una nueva funcionalidad

- Herramientas de gestión de casos de prueba: El tipo de herramientas de gestión de casos de prueba utilizado en los procesos ágiles puede formar parte de la herramienta de gestión del ciclo de vida de la aplicación o de la herramienta de gestión de tareas de todo el equipo
- Herramientas de preparación de datos de prueba y herramientas de generación de datos de prueba: Las herramientas que generan datos para poblar la base de datos de una aplicación son muy beneficiosas cuando se requieren muchos datos y

combinaciones de datos para probar la aplicación. Estas herramientas también pueden ayudar a redefinir la estructura de la base de datos a medida que el producto es objeto de cambios durante un proyecto ágil y refactorizar los scripts para generar los datos. Esto permite una rápida actualización de los datos de prueba a medida que suceden los cambios. Algunas herramientas de preparación de datos de prueba utilizan fuentes de datos de producción como materia prima y utilizan scripts para eliminar o anonimizar datos sensibles. Otras herramientas de preparación pueden ayudar a validar grandes entradas o salidas de datos

- Herramientas de carga de datos de prueba: Una vez generados los datos para las pruebas, deben cargarse en la aplicación. La entrada manual de datos a menudo lleva mucho tiempo y es proclive a cometer errores, no obstante hay herramientas de carga de datos disponibles para conseguir que el proceso sea fiable y eficiente. De hecho, muchas herramientas generadoras de datos llevan incorporado un componente de carga de datos. En otros casos, también puede recurrirse a la carga masiva a través de los sistemas de gestión de bases de datos
- Herramientas de ejecución de pruebas automatizadas: Hay herramientas de ejecución de pruebas que están más alineadas con las pruebas ágiles. Hay herramientas específicas de código abierto y comerciales que permiten soportar enfoques de probar primero, como son el desarrollo guiado por el comportamiento, el desarrollo guiado por pruebas y el desarrollo guiado por pruebas de aceptación. Estas herramientas permiten a los probadores y al personal de negocio expresar el comportamiento esperado del sistema en tablas o en lenguaje natural utilizando palabras clave
- Herramienta de pruebas exploratorias: Las herramientas que capturan y registran actividades llevadas a cabo en la aplicación durante una sesión de pruebas exploratorias son beneficiosas para el probador y el desarrollador, ya que registran las acciones realizadas. Estas herramientas resultan útiles cuando se detecta un defecto, ya que las acciones realizadas antes de que sucediera el fallo se han capturado y pueden utilizarse para reportar el defecto a los desarrolladores. Los pasos de registro llevados a cabo en una sesión de pruebas exploratorias pueden ser beneficiosos si la prueba se incluye en última instancia en el juego de pruebas de regresión automatizadas

4.6 Herramientas de Cloud Computing y virtualización

La virtualización permite a un recurso físico único (servidor) operar como muchos recursos independientes más pequeños. Cuando se utilizan máquinas virtuales o instancias en la nube, los equipos tienen un mayor número de servidores disponibles para desarrollo y pruebas. Esto puede ayudar a evitar retrasos asociados a la espera de servidores físicos. Provisionar un nuevo servidor o restaurar un servidor resulta más eficiente ya que la mayoría de las herramientas de virtualización llevan incorporada la capacidad de hacer instantáneas. Algunas herramientas de gestión de pruebas actualmente utilizan tecnologías de virtualización para realizar la instantánea de los servidores en el momento en que se detecta

un fallo, lo que permite a los probadores compartir la instantánea con los desarrolladores que están investigando el fallo.