Deloitte | 2022

Certificado del ISTQB®

Programa de Estudio de Nivel Básico

Extensión Ágil

Resumen



# Índice

**1. Desarrollo Ágil de Software**

[1.1 Fundamentos de Desarrollo Ágil de Software](#_1.1_Fundamentos_de)

1.1.1 El Desarrollo Ágil de Software y el Manifiesto Ágil

1.1.2 Enfoque de Equipo Completo

1.1.3 Retroalimentación Temprana y Frecuente

[1.2 Características de los Enfoques Ágiles](#_1.2_Características_de)

1.2.1 Enfoques de Desarrollo Ágil de Software

1.2.2 Creación Colaborativa de Historias de Usuario

1.2.3 Retrospectivas

1.2.4 Integración Continua

1.2.5 Planificación de Entregas e Iteraciones

**2. Principios, Prácticas y Procesos Fundamentales de Prueba Ágil**

[2.1 Las Diferencias entre Probar en Enfoques Tradicionales y Ágiles](#_2.1_Las_Diferencias)

2.1.1 Actividades de Prueba y Desarrollo

2.1.2 Productos de Trabajo de un Proyecto

2.1.3 Niveles de Prueba

2.1.4 Prueba y Gestión de la Configuración

2.1.5 Opciones de Organización para Pruebas Independientes

[2.2 Situación de la Prueba en Proyectos Ágiles](#_2.2_Situación_de)

2.2.1 Comunicar el Estado de la Prueba, el Avance y la Calidad del Producto

2.2.2 Gestión del Riesgo de Regresión con Casos de Prueba Manuales y Automatizados en Evolución

[2.3 Rol y Competencias de un Probador en un Equipo Ágil](#_2.3_Rol_y)

2.3.1 Competencias del Probador Ágil

2.3.2 El Rol de un Probador en un Equipo Ágil

**3. Métodos, Técnicas y Herramientas de Prueba Ágiles**

[3.1 Método de Prueba Ágil](#_3.1_Métodos_de)

3.1.1 Tipos de desarrollo

3.1.2 La Pirámide de Prueba

3.1.3 Cuadrantes de Prueba

[3.2 Evaluación de los Riesgos de Calidad y Estimación del Esfuerzo de Prueba](#_3.2_Evaluación_de)

3.2.1 Evaluación de los Riesgos de Calidad en Proyectos Ágil

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

[3.3 Técnicas en Proyectos Ágiles](#_3.3_Técnicas_en)

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

3.3.2 Aplicación del Desarrollo Guiado por Pruebas de Aceptación

3.3.3 Diseño de Pruebas de Caja Negra Funcionales y No Funcionales

3.3.4 Prueba Exploratoria

[3.4 Herramientas en Proyectos Ágiles](#_3.4_Herramientas_en)

3.4.1 Herramientas de Gestión y Seguimientos de Tareas

3.4.2 Herramientas de Comunicación e Intercambio de Información

3.4.3 Herramientas de Construcción y Distribución de Software

3.4.4 Herramientas de Gestión de la Configuración

3.4.5 Herramientas de Diseño de Pruebas, Implementación y Ejecución

3.4.6 Informática en la Nube y Herramientas de Virtualización

1. Desarrollo Ágil de Software

# 1.1 Fundamentos de Desarrollo Ágil de Software

## El Desarrollo Ágil de Software y el Manifiesto Ágil

El Manifiesto Ágil contiene cuatro enunciados de valores:

* Individuos e interacciones sobre procesos y herramientas
* Software funcionando sobre documentación extensiva
* Colaboración con el cliente sobre negociación contractual
* Respuesta ante el cambio sobre seguir un plan

### Individuos e interacciones

El desarrollo ágil está muy centrado en las personas. Los equipos de personas construyen software, y es a través de la comunicación e interacción continuas, en lugar de depender de herramientas o procesos, que los equipos pueden trabajar con mayor eficacia

### Software funcionando

Desde el punto de vista del cliente, software funcionando es mucho más útil y valioso que documentación excesivamente detallada y ofrece la oportunidad de proporcionar al equipo de desarrollo una retroalimentación rápida.

Por lo tanto, el desarrollo ágil es especialmente útil en entornos de negocio que cambian rápidamente, en los que los problemas y/o las soluciones no están claros o en los que el área de negocio desea innovar en nuevos dominios de problemas.

### Colaboración con el cliente

Los clientes suelen tener grandes dificultades para especificar el sistema que necesitan. Colaborar directamente con el cliente mejora la posibilidad de comprender exactamente lo que éste requiere

### Respuesta ante el servicio

El cambio es inevitable en los proyectos de software. El proceso de desarrollo debe dar cabida a estos factores. Por lo tanto, tener flexibilidad en las prácticas de trabajo para aceptar el cambio es más importante que simplemente adherirse rígidamente a un plan.

### Principios

Los valores clave del Manifiesto Ágil se recogen en doce principios:

* Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
* Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
* Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
* Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
* Los proyectos se construyen en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
* El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
* El software funcionando es la medida principal de progreso.
* Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
* La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.
* La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
* Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
* A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

## Enfoque de Equipo Completo

El enfoque de equipo completo implica la participación de todas las personas con los conocimientos y competencias necesarios para garantizar el éxito del proyecto. El equipo incluye representantes del cliente y otros implicados del negocio que determinan las prestaciones del producto

El enfoque de equipo completo se apoya en las reuniones de pie diarias en las que participan todos los miembros del equipo, en las que se comunica el progreso del trabajo y se pone de manifiesto cualquier impedimento para avanzar. El enfoque de equipo completo promueve una dinámica de equipo más eficaz y eficiente.

El uso de un enfoque de equipo completo para el desarrollo del producto es una de las principales ventajas del desarrollo ágil. Sus beneficios incluyen:

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

Todo el equipo es responsable de la calidad en los proyectos ágiles. La esencia del enfoque de equipo completo reside en que los probadores, los desarrolladores y los representantes de negocio trabajen juntos en cada paso del proceso de desarrollo.

Todo el equipo participa en las consultas o reuniones en las que se presentan, analizan o estiman las prestaciones del producto.

### Retroalimentación Temprana y Frecuente

Los proyectos ágiles tienen iteraciones cortas que permiten al equipo del proyecto recibir una retroalimentación continua y temprana sobre la calidad del producto a lo largo del ciclo de vida de desarrollo.

Cuando se utilizan enfoques de desarrollo secuencial, el cliente no suele ver el producto hasta que el proyecto está casi terminado. En ese momento, suele ser demasiado tarde para que el equipo de desarrollo aborde con eficacia cualquier problema que pueda tener el cliente. Al obtener retroalimentación frecuente por parte del cliente a medida que avanza el proyecto, los equipos ágiles pueden incorporar la mayoría de los nuevos cambios en el proceso de desarrollo del producto.

Los beneficios de una retroalimentación temprana y frecuente incluyen:

* Evitar malentendidos sobre los requisitos, que podrían no detectarse hasta más adelante en el ciclo de desarrollo, cuando son más costosos de solucionar.
* Aclarar las solicitudes de prestaciones de los clientes, haciéndolas disponibles para el uso de estos con antelación. De este modo, el producto refleja mejor lo que quiere el cliente.
* Descubrir (a través de la integración continua), aislar y resolver los problemas de calidad de forma temprana.
* Aportar información al equipo ágil sobre su productividad y capacidad para realizar entregas.
* Promover un impulso consistente al proyecto

# 1.2 Características de los Enfoques Ágiles

## 1.2.1 Enfoques de Desarrollo Ágil de Software

Existen varios enfoques ágiles, cada uno de los cuales implementa los valores y principios del Manifiesto ágil de diferentes maneras. En este programa de estudio, se consideran tres representantes de los enfoques ágiles: Programación extrema (XP), Scrum y Kanban.

### Programación Extrema

La Programación Extrema (XP), introducida originalmente por Kent Beck, es un enfoque ágil para el desarrollo de software descrito por ciertos valores, principios y prácticas de desarrollo.

* XP adopta cinco valores para guiar el desarrollo: comunicación, simplicidad, retroalimentación, valor y respeto.
* XP describe un conjunto de principios como directrices adicionales: humanidad, economía, beneficio mutuo, auto-similitud, mejora, diversidad, reflexión, flujo, oportunidad, redundancia, fallo, calidad, pasos de bebé y responsabilidad aceptada.
* XP describe trece prácticas principales: sentarse juntos, equipo completo, espacio de trabajo informativo, trabajo energizado, programación en pareja, historias, ciclo semanal, ciclo trimestral, holgura, construcción en diez minutos, integración continua, programar probando primero y diseño incremental.

### Scrum

Scrum es un marco de gestión ágil que contiene los siguientes instrumentos y prácticas constitutivos:

* Esprints: Scrum divide un proyecto en iteraciones (denominadas esprints) de duración fija (normalmente de dos a cuatro semanas)
* Incremento de Producto: Cada esprint da como resultado un producto que potencialmente se puede entregar o enviar (llamado incremento)
* Trabajo Acumulado del Producto: El propietario de producto gestiona una lista priorizada de elementos del producto planificados
* Trabajo Acumulado del Esprint: Al comienzo de cada Sprint, el equipo Scrum selecciona un conjunto de elementos de máxima prioridad a partir del trabajo acumulado del producto
* Definición de Hecho: Para asegurarse de que hay un producto potencialmente entregable al final de cada esprint, el equipo Scrum discute y define los criterios adecuados para la entrega del esprint
* Acotamiento del Tiempo: Sólo las tareas, requisitos o prestaciones que el equipo espera terminar dentro del esprint forman parte del trabajo acumulado del esprint
* Transparencia: El equipo de desarrollo informa y actualiza diariamente el estado del esprint en una reunión denominada scrum diario.

Scrum define tres roles:

* Scrum Máster: se asegura de que las prácticas y reglas de Scrum se implementen y se cumplan, y resuelve cualquier incumplimiento, problemas de recursos u otros impedimentos que podrían impedir que el equipo siga las prácticas y reglas. Esta persona no es el líder del equipo, sino un entrenador.
* Propietario de producto: representa al cliente y genera, mantiene y prioriza el trabajo acumulado del producto. Esta persona no es el líder del equipo.
* Equipo de Desarrollo: desarrolla y prueba el producto. El equipo se organiza por sí mismo: No hay un jefe de equipo, por lo que el equipo toma las decisiones. El equipo también es interfuncional

### Kanban

Kanban es un enfoque de gestión que a veces se utiliza en los proyectos ágiles. El objetivo general es visualizar y optimizar el flujo de trabajo dentro de una cadena de valor añadido. Kanban utiliza tres instrumentos:

* Tablero Kanban: La cadena de valor que se gestiona se visualiza mediante un tablero Kanban. Cada columna muestra una estación, que es un conjunto de actividades relacionadas
* Limitación del Trabajo en Curso: La cantidad de tareas activas en paralelo está estrictamente limitada. Esto se controla mediante el número máximo de tickets permitido para una estación y/o globalmente para el tablero
* Plazo de Ejecución: Kanban se utiliza para optimizar el flujo continuo de tareas minimizando el plazo de ejecución (medio) de todo el flujo de valor

## 1.2.2 Creación Colaborativa de Historias de Usuario

Las historias de usuario deben abordar tanto las características funcionales como las no funcionales. Cada historia incluye criterios de aceptación para estas características. Estos criterios deben definirse en colaboración entre los representantes de negocio, los desarrolladores y los probadores-

Un equipo ágil considera que una tarea está terminada cuando se han satisfecho un conjunto de criterios de aceptación.

La autoría colaborativa de la historia de usuario puede utilizar técnicas como la tormenta de ideas y los mapas mentales. El probador puede utilizar la técnica INVEST (Independiente, Negociable, Valioso, Estimable, Pequeño, Capacidad de Ser Probado):

Según el concepto 3C, una historia de usuario es la conjunción de tres elementos:

* Tarjeta: La tarjeta es el soporte físico que describe una historia de usuario. Identifica el requisito, su criticidad, la duración prevista del desarrollo y de las pruebas, y los criterios de aceptación de esa historia. La descripción debe ser precisa, ya que se utilizará en la lista de trabajo acumulado de producto.
* Conversación: La conversación explica cómo se utilizará el software. La conversación puede estar documentada o ser verbal
* Confirmación: Los criterios de aceptación, discutidos en la conversación, se utilizan para confirmar que la historia está hecha. Estos criterios de aceptación pueden abarcar varias historias de usuario. Deben utilizarse pruebas tanto positivas como negativas para cubrir los criterios

### 1.2.3 Retrospectivas

En el desarrollo ágil, una retrospectiva es una reunión que se celebra al final de cada iteración para discutir sobre lo que tuvo éxito, lo que podría mejorarse y cómo incorporar las mejoras y conservar los éxitos en futuras iteraciones. Las retrospectivas abarcan temas como el proceso, las personas, las organizaciones, las relaciones y las herramientas.

Las retrospectivas pueden dar lugar a decisiones de mejora relacionadas con la prueba, centradas en la efectividad de la prueba, la productividad de la prueba, la calidad de los casos de prueba y la satisfacción del equipo.

El momento y la organización de la retrospectiva dependen del método ágil concreto que se siga.

Los probadores deberían desempeñar un papel importante en las retrospectivas. Los probadores forman parte del equipo y aportan su perspectiva única. Todos los miembros del equipo, probadores y no probadores pueden aportar su contribución tanto a las actividades de prueba como a las que no lo son.

### 1.2.4 Integración Continua

La entrega de un incremento de producto requiere un software fiable, que funcione y esté integrado al final de cada esprint. La integración continua aborda este reto fusionando todos los cambios realizados en el software e integrando todos los componentes modificados regularmente, al menos una vez al día.

Un proceso de integración continua consta de las siguientes actividades automatizadas:

* Análisis estático de código: llevar a cabo el análisis estático de código e informar los resultados.
* Compilación: compilar y enlazar el código, generando los archivos ejecutables.
* Prueba unitaria: ejecutar las pruebas unitarias, comprobar la cobertura de código e informar los resultados de la prueba.
* Despliegue: instalar el producto construido en un entorno de prueba.
* Prueba de integración: ejecutar las pruebas de integración e informar los resultados.
* Informe (panel de control): publicar el estado de todas estas actividades en un lugar visible para el público o enviar por correo electrónico el estado al equipo.

La integración continua permite a los probadores ágiles realizar pruebas automatizadas con regularidad, en algunos casos como parte del propio proceso de integración continua, y enviar una retroalimentación rápida al equipo sobre la calidad del código

Además de las pruebas automatizadas, las organizaciones que utilizan la integración continua suelen emplear herramientas de construcción para implementar el control de la calidad continuo. Además de ejecutar pruebas unitarias y de integración, estas herramientas pueden ejecutar pruebas estáticas y dinámicas adicionales, medir y perfilar el rendimiento, extraer y formatear la documentación del código fuente y facilitar los procesos manuales de control de calidad.

La integración continua puede aportar las siguientes ventajas:

* Permite detectar antes y facilitar el análisis de la causa raíz de los problemas de integración y los cambios conflictivos.
* Aporta al equipo de desarrollo retroalimentación periódica sobre si el código está funcionando.
* Mantiene la versión del software que se está probando a un día de la versión que se está desarrollando.
* Reduce el riesgo de regresión asociado a la refactorización del código del desarrollador gracias a la rápida repetición de la prueba de la base de código después de cada pequeño conjunto de cambios.
* Aporta la confianza de que el trabajo de desarrollo de cada día se basa en una base sólida.
* Hace visible el progreso hacia la compleción del incremento de producto, animando a los desarrolladores y probadores.
* Elimina los riesgos de calendario asociados a la integración big bang.
* Aporta una disponibilidad constante de software ejecutable durante todo el esprint para fines de prueba, demostración o educación.
* Reduce las actividades repetitivas de la prueba manual.
* Proporciona una retroalimentación rápida sobre las decisiones tomadas para mejorar la calidad y las pruebas.

### 1.2.5 Planificación de Entregas e Iteraciones

La planificación es una actividad continua, y este también es el caso de los ciclos de vida ágiles. En los ciclos de vida ágiles, se producen dos tipos de planificación, la planificación de la entrega y la planificación de la iteración.

La planificación de la entrega se anticipa al lanzamiento de un producto, a menudo unos meses antes del inicio de un proyecto. La planificación de la entrega define y redefine la lista de trabajo acumulado del producto

Los probadores participan en la planificación de la entrega y añaden valor especialmente en las siguientes actividades:

* Definir historias de usuario que puedan ser probadas, incluidos los criterios de aceptación.
* Participar en los análisis del riesgo de proyecto y de calidad.
* Estimar el esfuerzo de prueba asociado a las historias de usuario.
* Definir los niveles de prueba necesarios.
* Planificar la prueba para la entrega

En la planificación de la iteración, el equipo selecciona las historias de usuario de la lista de trabajo acumulado priorizada, elabora las historias de usuario, realiza un análisis del riesgo para las historias de usuario y estima el trabajo necesario para cada historia de usuario.

El número de historias seleccionadas se basa en la velocidad establecida del equipo y en el tamaño estimado de las historias de usuario seleccionadas.

Los probadores participan en la planificación de la iteración y añaden valor especialmente en las siguientes actividades:

* Participando en el análisis del riesgo detallado de las historias de usuario.
* Determinar la capacidad de ser probada de las historias de usuario.
* Crear pruebas de aceptación para las historias de usuario.
* Descomponer las historias de usuario en tareas (en particular, las tareas de prueba).
* Estimar el esfuerzo de prueba para todas las tareas de prueba.
* Identificar los aspectos funcionales y no funcionales del sistema que se va a probar.
* Apoyar y participar en la automatización de la prueba en diversos niveles de prueba.

Los planes de entrega pueden cambiar a medida que avanza el proyecto, incluyendo cambios en las historias de usuario individuales en la lista de trabajo acumulado del producto. Estos cambios pueden ser provocados por factores internos o externos.

Estos cambios pueden suponer un reto para los probadores. Los probadores deben entender el panorama general de la entrega a efectos de la planificación de la prueba, y deben tener una base de prueba adecuada.

Las planificaciones de la entrega y de la iteración deben abordar la planificación de prueba, así como la planificación de las actividades de desarrollo. Entre las cuestiones particulares relacionadas con las pruebas que deben abordarse se encuentran:

* El alcance de la prueba, la extensión de la prueba para aquellas áreas en el alcance, los objetivos de la prueba y las razones de estas decisiones.
* Los miembros del equipo que llevarán a cabo las actividades de prueba.
* El entorno de prueba y los datos de prueba necesarios
* El momento, la secuencia, las dependencias y los prerrequisitos para las actividades de prueba funcionales y no funcionales
* Los riesgos de proyecto y de calidad que deben abordarse

Además, el esfuerzo de estimación del equipo más grande debe incluir la consideración del tiempo y el esfuerzo necesarios para completar las actividades de prueba necesarias.

2. Principios, Prácticas y Procesos Fundamentales de Prueba Ágil

# 2.1 Las Diferencias entre Probar en Enfoques Tradicionales y Ágiles

Las actividades de prueba están relacionadas con las actividades de desarrollo y, por tanto, la prueba varía en los distintos ciclos de vida. Los probadores deben comprender las diferencias entre la prueba en los modelos de ciclo de vida tradicionales y los ciclos de vida ágiles para poder trabajar con eficacia y eficiencia.

## 2.1.1 Actividades de Prueba y Desarrollo

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 todo un papel en la prueba, como en los ciclos de vida tradicionales. Los desarrolladores realizan pruebas unitarias a medida que desarrollan las prestaciones a partir de las historias de usuario. A continuación, los probadores prueban esas prestaciones. Los implicados del negocio también prueban las historias durante la implementación. Los implicados del negocio pueden utilizar casos de prueba escritos, pero también pueden simplemente experimentar y utilizar la prestación para proporcionar una retroalimentación rápida al equipo de desarrollo.

Cuando se utiliza la prueba basada en el riesgo como una de las estrategias de prueba, se produce un análisis del riesgo de alto nivel durante la planificación de la entrega, y los probadores suelen dirigir ese análisis.

En algunas prácticas ágiles se utiliza el trabajo en pareja. El trabajo en pareja puede implicar que los probadores trabajen juntos de dos en dos para probar una prestación. El trabajo en pareja también puede implicar que un probador trabaje en colaboración con un desarrollador para desarrollar y probar una prestación.

Los probadores también pueden actuar como entrenadores en la prueba y calidad dentro del equipo, compartiendo los conocimientos sobre la prueba y apoyando el trabajo de aseguramiento de la calidad dentro del equipo.

La automatización de la prueba en todos los niveles tiene lugar en muchos equipos ágiles, y esto puede significar que los probadores pasan tiempo creando, ejecutando, monitorizando y manteniendo las pruebas y resultados automatizados.

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.

## 2.1.2 Productos de Trabajo de un Proyecto

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

1. Productos de trabajo orientados al negocio que describen lo que se necesita (por ejemplo, especificaciones de requisitos) y cómo utilizarlo
2. Productos de trabajo de desarrollo que describen cómo se construye el sistema (por ejemplo, diagramas entidad-relación de la base de datos), que realmente implementan el sistema o que evalúan piezas individuales de código
3. Productos de trabajo de prueba que describen cómo se prueba el sistema (por ejemplo, estrategias y planes de prueba), que realmente prueban el sistema o que presentan los resultados de las pruebas

En un proyecto ágil típico, es una práctica habitual evitar la producción de grandes cantidades de documentación. En su lugar, se centra más en tener un software que funcione, junto con pruebas automatizadas que demuestren la conformidad con los requisitos.

Los productos de trabajo típicos orientados al negocio en los proyectos ágiles incluyen historias de usuario y criterios de aceptación. 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.

Las colecciones más grandes de características relacionadas pueden denominarse "épicas". Las épicas pueden incluir historias de usuario para diferentes equipos de desarrollo.

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.

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.

En algunas implementaciones ágiles, se requiere una mayor formalización de estos productos de trabajo.

## 2.1.3 Niveles de Prueba

Los niveles de prueba son actividades de prueba que están relacionadas de forma lógica, a menudo por la madurez o completitud del elemento sujeto a prueba.

En los modelos de ciclo de vida secuencial, los niveles de prueba suelen definirse de forma que los criterios de salida de un nivel forman parte de los criterios de entrada del siguiente.

Durante una iteración, cualquier historia de usuario determinada suele avanzar secuencialmente a través de las siguientes actividades de prueba:

* Prueba unitaria, normalmente llevada a cabo por el desarrollador.
* Prueba de aceptación de prestaciones, que a veces se divide en dos actividades:
  + Prueba de verificación de prestación, que suele estar automatizada, puede ser realizada por desarrolladores o probadores
  + Prueba de validación de la prestación, que suele ser manual y puede involucrar a los desarrolladores, probadores y a los implicados del negocio

Además, suele haber un proceso paralelo de pruebas de regresión que se produce a lo largo de la iteración. Esto implica volver a ejecutar las pruebas unitarias automatizadas y las pruebas de verificación de las prestaciones de la iteración actual y de las anteriores, normalmente a través de un marco de integración continua.

En algunos proyectos ágiles, puede haber un nivel de prueba del sistema, que se inicia una vez que la primera historia de usuario está lista para dicha prueba.

## 2.1.4 Prueba y Gestión de la configuración

Los proyectos ágiles suelen implicar un uso intensivo de herramientas automatizadas para desarrollar, probar y gestionar el desarrollo de software. Los desarrolladores utilizan herramientas para el análisis estático, las pruebas unitarias y la cobertura de código.

Estas pruebas automatizadas también pueden incluir pruebas funcionales a nivel de integración y de sistema.

Uno de los objetivos de las pruebas automatizadas es confirmar que la construcción funciona y es instalable. Si alguna prueba automatizada falla, el equipo debe corregir el defecto subyacente a tiempo para el siguiente registro de entrada del código.

## 2.1.5 Opciones de Organización para Pruebas Independientes

Los probadores independientes suelen ser más eficaces a la hora de encontrar defectos. En algunos equipos ágiles, los desarrolladores crean muchas de las pruebas en forma de pruebas automatizadas. Uno o varios probadores pueden estar integrados en el equipo y realizar muchas de las tareas de prueba.

Otros equipos ágiles mantienen equipos de pruebas totalmente independientes y separados, y asignan probadores a demanda durante los últimos días de cada esprint.

Una tercera opción es contar con un equipo de pruebas independiente y separado en el que los probadores sean asignados a equipos ágiles a largo plazo, al principio del proyecto, lo que les permite mantener su independencia al tiempo que adquieren un buen conocimiento del producto y una sólida relación con los demás miembros del equipo.

# 2.2 Situación de la Prueba en Proyectos Ágiles

Los cambios se producen rápidamente en los proyectos ágiles. Este cambio significa que el estado de las pruebas, el avance de las mismas y la calidad del producto evolucionan constantemente, y los probadores deben idear formas de hacer llegar esa información al equipo para que puedan tomar decisiones que les permitan mantener el rumbo para completar con éxito cada iteración.

## 2.2.1 Comunicar el Estado de la Prueba, el Avance y la Calidad del Producto

Los equipos ágiles avanzan en la medida en que tengan un software funcionando al final de cada iteración. Para determinar cuándo el equipo tendrá software funcionando, necesitan monitorizar el avance de todos los elementos de trabajo en la iteración y la entrega.

Los equipos pueden utilizar gráficos de quemado para hacer un seguimiento del avance en toda la entrega y dentro de cada iteración. Un gráfico de quemado representa la cantidad de trabajo que queda por hacer frente al tiempo asignado a la entrega o iteración.

Para ofrecer una representación visual instantánea y detallada del estado actual de todo el equipo se puede utilizar tableros de tareas ágiles. Las tarjetas de historia, las tareas de desarrollo, las tareas de prueba y otras tareas creadas durante la planificación de la iteración se plasman en el tablero de tareas, a menudo utilizando tarjetas coordinadas por colores para determinar el tipo de tarea.

Las tareas de prueba del tablero de tareas están relacionadas con los criterios de aceptación definidos para las historias de usuario.

Todo el equipo revisa el estado del tablero de tareas con regularidad, a menudo durante las reuniones de pie diarias, para asegurarse de que las tareas se mueven por el tablero a un ritmo aceptable.

En esta reunión, comunican su estado actual. La agenda para cada miembro es:

* ¿Qué ha completado desde la última reunión?
* ¿Qué tiene previsto completar para la próxima reunión?
* ¿Qué se está interponiendo en su camino?

Como en cualquier ciclo de vida, las métricas recopiladas e informadas deben ser relevantes y ayudar a la toma de decisiones.

## 2.2.2 Gestión del Riesgo de Regresión con Casos de Prueba Manuales y Automatizados en Evolución

En un proyecto ágil, a medida que se completa cada iteración, el producto crece. Por tanto, el alcance de las pruebas también aumenta.

El riesgo de introducir una regresión en el desarrollo ágil es alto debido al amplio batido de código (líneas de código añadidas, modificadas o eliminadas de una versión a otra). Dado que responder al cambio es un principio ágil clave, también se pueden realizar cambios en las prestaciones entregadas previamente para satisfacer las necesidades del negocio. Para mantener la velocidad sin incurrir en una gran cantidad de deuda técnica, es fundamental que los equipos inviertan en la automatización de la prueba en todos los niveles de prueba lo antes posible.

Dado que la repetición completa de todas las pruebas rara vez es posible, especialmente en los proyectos ágiles con plazos ajustados, los probadores deben asignar tiempo en cada iteración para revisar los casos de prueba manuales y automatizados de las iteraciones anteriores y actuales para seleccionar los casos de prueba que pueden ser candidatos para el juego de prueba de regresión.

Al revisar los casos de prueba, los probadores deben considerar la adecuación para la automatización. El equipo debe automatizar el mayor número posible de pruebas de las iteraciones anteriores y actuales.

Es fundamental que los probadores tengan la capacidad de identificar y actualizar rápidamente los casos de prueba de iteraciones y/o entregas anteriores que se ven afectados por los cambios realizados en la iteración en curso.

Los plazos más cortos para la prueba y el cambio constante en cada iteración aumentarán el impacto de las malas prácticas de diseño e implementación de pruebas.

El uso de la automatización de la prueba, en todos los niveles de prueba, permite a los equipos ágiles proporcionar una retroalimentación rápida sobre la calidad de producto.

Las pruebas unitarias automatizadas se ejecutan antes de registrar el código fuente en la línea principal del sistema de gestión de la configuración para garantizar que los cambios en el código no rompan la construcción del software.

Las pruebas de aceptación automatizadas se ejecutan regularmente como parte de la construcción del sistema completo de integración continua. Estas pruebas se ejecutan contra una construcción completa del sistema al menos a diario.

Se pueden ejecutar pruebas automatizadas de forma continua contra el sistema.

Estas pruebas se conocen comúnmente como pruebas de verificación de la construcción. Los resultados de las pruebas de verificación de la construcción proporcionarán una retroalimentación instantánea sobre el software después de su despliegue, para que los equipos no pierdan tiempo probando una construcción inestable.

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

* Generación de datos de prueba.
* Carga de los datos de prueba en los sistemas.
* Despliegue de construcciones en los entornos de prueba.
* Restauración de un entorno de prueba (por ejemplo, la base de datos o los archivos de datos del sitio web) a una línea de base.
* Comparación de las salidas de datos

La automatización de estas tareas reduce la sobrecarga y permite al equipo dedicar tiempo al desarrollo y prueba de nuevas prestaciones.

# 2.3 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.

## 2.3.1 Competencias del Probador Ágil

Los probadores ágiles deben tener todas las competencias mencionadas en el programa de estudio de nivel básico. 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 (en lugar de basarse totalmente en las especificaciones escritas).
* 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.

## 2.3.2 El Rol de un Probador en un Equipo Ágil

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. 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

3. Métodos, Técnicas y Herramientas de Prueba Ágiles

# 3.1 Métodos de Prueba Ágil

Hay ciertas prácticas de prueba que pueden seguirse en todo proyecto de desarrollo (ágil o no) para producir productos de calidad. Entre ellas se incluye la redacción de pruebas con antelación para expresar el comportamiento adecuado.

## 3.1.1 Tipos de desarrollo

El desarrollo guiado por pruebas, el desarrollo guiado por pruebas de aceptación y el desarrollo guiado por el comportamiento son tres técnicas complementarias que se utilizan en equipos ágiles para llevar a cabo las pruebas en los distintos niveles de prueba. Cada técnica es un ejemplo de un principio fundamental de la prueba, el beneficio de las pruebas tempranas y las actividades de control de calidad, ya que las pruebas se definen antes de escribir el código.

### Desarrollo Guiado por Pruebas

El desarrollo guiado por pruebas se utiliza para desarrollar código guiado por casos de prueba automatizados. El proceso para el desarrollo guiado por pruebas es:

* Se añade una prueba que capture el concepto del programador sobre el funcionamiento deseado de un pequeño fragmento de código.
* Se ejecuta la prueba, que debería fallar ya que el código no existe.
* Se escribe el código y se ejecuta la prueba en un bucle cerrado hasta que la prueba pase.
* Se refactoriza el código después de que la prueba haya sido superada, y 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

### 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 desarrollo 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.

### 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. Como las pruebas se basan en el comportamiento exhibido del software, las pruebas suelen ser más fáciles de entender para los demás miembros del equipo y los implicados.

## 3.1.2 La Pirámide de Prueba

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

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

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

## 3.1.3 Cuadrantes de Prueba

Los cuadrantes de prueba alinean los niveles de prueba con los tipos de prueba adecuados en la metodología ágil. El modelo de cuadrantes de prueba, y sus variantes, ayuda a garantizar que todos los tipos y niveles de prueba importantes se incluyan en el ciclo de vida de desarrollo.

Las pruebas pueden ser totalmente manuales, totalmente automatizadas, una combinación de manuales y automatizadas, o manuales pero apoyadas por herramientas. Los cuatro cuadrantes son los siguientes:

* 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. Estas pruebas comprueban los criterios de aceptación y pueden ser manuales o 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. Estas pruebas suelen ser manuales y están orientadas al usuario.
* 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. Estas pruebas suelen estar automatizadas.

## 3.1.4 El Rol de un Probador

Esta subsección examina específicamente el rol de un probador en un proyecto que sigue un ciclo de vida Scrum:

### Trabajo en Equipo

Las siguientes son las buenas prácticas organizativas y de comportamiento en los equipos de Scrum:

* Interdisciplinario: Cada miembro del equipo aporta un conjunto diferente de competencias al equipo
* Autoorganizado: El equipo puede estar formado sólo por desarrolladores, lo ideal sería que hubiera uno o más probadores
* Ubicación común: Los probadores se sientan junto con los desarrolladores y el propietario de producto.
* Colaborativo: Los probadores colaboran con los miembros de su equipo, con los implicados, con el propietario de producto y con el Scrum Master.
* Empoderado: Las decisiones técnicas relativas al diseño y las pruebas las toma el equipo en su conjunto, en colaboración con el propietario del producto y otros equipos si es necesario.
* Comprometido: El probador se compromete a cuestionar y evaluar el comportamiento y las características del producto con respecto a las expectativas y necesidades de los clientes y usuarios
* Transparente: Los avances en el desarrollo y la prueba son visibles en el tablero de tareas ágil
* Creíble: El probador debe garantizar la credibilidad de la estrategia de prueba, su implementación y ejecución, de lo contrario los implicados no confiarán en los resultados de la prueba.
* Abierto a la retroalimentación: La retroalimentación es un aspecto importante para tener éxito en cualquier proyecto
* Resiliente: La prueba debe ser capaz de responder al cambio, como todas las demás actividades en los proyectos ágiles.

### Esprint Cero

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

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

### Integración

En los proyectos ágiles, el objetivo es entregar valor al cliente de forma continua. Para permitirlo, la estrategia de integración debe considerar tanto el diseño como la prueba. Para permitir una estrategia de prueba continua de la funcionalidad y las características entregadas, es importante identificar todas las dependencias entre las funciones y características subyacentes.

### Planificación de la Prueba

Dado que las pruebas están totalmente integradas en el equipo ágil, la planificación de la prueba debe comenzar durante la sesión de planificación de la entrega y actualizarse durante cada esprint.

La planificación del esprint da como resultado un conjunto de tareas que se colocan en el tablero de tareas, donde cada tarea debe tener una duración de uno o dos días de trabajo.

### Prácticas de Prueba Ágil

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

* Trabajo en pareja: Dos miembros del equipo se sientan juntos en un puesto de trabajo para realizar una prueba u otra tarea del esprint.
* Diseño de prueba incremental: Los casos de prueba y los contratos se construyen gradualmente a partir de las historias de usuario y otras bases de prueba
* Mapa Mental: Los mapas mentales son una herramienta útil a la hora de probar.

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

Un objetivo típico de la prueba en todos los proyectos, ágiles o tradicionales, es reducir el riesgo de problemas de calidad de producto a un nivel aceptable antes de su entrega.

## 3.2.1 Evaluación de los Riesgos de Calidad en Proyectos Ágiles

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

Los probadores de los equipos ágiles pueden utilizar la identificación del riesgo, el análisis y las estrategias de mitigación del riesgo para ayudar a determinar un número aceptable de casos de prueba a ejecutar, aunque muchas limitaciones y variables que interactúan pueden requerir compromisos.

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

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

Estas tareas pueden priorizarse en parte en función del nivel de riesgo de calidad asociado a ellas. Las tareas asociadas con riesgos más altos deberían empezar antes e implicar un mayor esfuerzo de prueba

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

1. Reunir a los miembros del equipo ágil, incluido el probador o probadores.
2. Hacer una lista de todos los elementos de la lista de trabajo acumulado 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 pertinentes.
4. Evaluar cada riesgo identificado, lo que incluye dos actividades: categorizar el riesgo y determinar su nivel de riesgo basándose en el impacto y la probabilidad de defectos.
5. Determinar el alcance de la prueba de forma proporcional al nivel de riesgo.
6. Seleccionar la(s) técnica(s) de prueba adecuada(s) para mitigar cada riesgo, basándose en el riesgo, el nivel de riesgo y la característica de calidad pertinente.

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

Durante la planificación de la entrega, el equipo ágil estima el esfuerzo necesario para completar la entrega. La estimación aborda también el esfuerzo de prueba. Una técnica de estimación habitual en los proyectos ágiles es el póker de planificación, una técnica basada en el consenso.

Los valores representan el número de puntos de historia, días de esfuerzo u otras unidades en las que el equipo estima.

Una estimación elevada suele significar que la historia no se entiende bien o que debe dividirse en varias historias más pequeñas.

Los estimadores discuten la prestación y hacen preguntas al propietario de producto si es necesario. Por lo tanto, es aconsejable incluir el nivel de riesgo de un elemento de la lista de trabajo acumulado, además de la prioridad especificada por el propietario del producto.

# 3.3 Técnicas en Proyectos Ágiles

Muchas de las técnicas de prueba y los niveles de prueba que se aplican a los proyectos tradicionales también pueden aplicarse a los proyectos ágiles. Sin embargo, para los proyectos ágiles hay que tener en cuenta algunas consideraciones específicas y variaciones en las técnicas de prueba, las terminologías y la documentación.

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

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

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

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

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

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

### Niveles de Prueba

Cada nivel de prueba tiene su propia definición de hecho. La siguiente lista ofrece ejemplos que pueden ser relevantes para los diferentes niveles de prueba.

* Prueba Unitaria
  + 100% de cobertura de decisión siempre que sea posible, con revisiones minuciosas de cualquier camino inviable
  + Análisis estático realizado en todo el código
  + Ningún defecto importante sin resolver
  + Ninguna deuda técnica inaceptable conocida en el diseño y el código
  + Todo el código, pruebas unitarias y resultados de pruebas unitarias revisados.
  + Todas las pruebas unitarias automatizadas.
  + Las características importantes están dentro de los límites acordados
* Prueba de Integración
  + Todos los requisitos funcionales probados, incluidas las pruebas positivas y negativas, con el número de pruebas basado en el tamaño, la complejidad y los riesgos.
  + Todas las interfaces entre unidades probadas.
  + Todos los riesgos de calidad cubiertos según el alcance acordado de la prueba.
  + Ningún defecto importante sin resolver (priorizado según el riesgo e importancia).
  + Todos los defectos detectados se comunican.
  + Todas las pruebas de regresión automatizadas, cuando sea posible, con todas las pruebas automatizadas almacenadas en un repositorio común.
* Prueba de Sistema
  + Pruebas de extremo a extremo de las historias de usuario, las prestaciones y las funciones.
  + Todas las personas usuarias cubiertas.
  + Cubiertas las características de calidad más importantes del sistema
  + Todos los riesgos de calidad cubiertos según el alcance acordado de la prueba.
  + Todas las pruebas de regresión automatizadas, cuando sea posible, con todas las pruebas automatizadas almacenadas en un repositorio común.
  + Todos los defectos encontrados se informan y posiblemente se solucionan.
  + No hay defectos importantes sin resolver

### Historia de Usuario

La definición de hecho para las historias de usuario puede estar determinada por los siguientes criterios:

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

### Prestación

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

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

### Iteración

La definición de hecho para la iteración puede incluir lo siguiente:

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

### Entrega

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

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

## 3.3.2 Aplicación del Desarrollo Guiado por Pruebas de Aceptación

El desarrollo guiado por pruebas de aceptación es un enfoque que da prioridad a las pruebas. Los casos de prueba se crean antes de implementar la historia de usuario. Los casos de prueba son creados por el equipo ágil, que incluye al desarrollador, al probador y a los representantes de negocio y pueden ser manuales o automatizados.

El siguiente paso es crear las pruebas. Esto puede hacerlo el equipo en conjunto o el probador individualmente. En cualquier caso, una persona independiente, como un representante de negocio, valida las pruebas. Las pruebas son ejemplos que describen las características específicas de la historia de usuario.

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

En la prueba Ágil, los probadores crean muchas pruebas de forma simultánea a las actividades de programación de los desarrolladores. Al igual que los desarrolladores programan basándose en las historias de usuario y los criterios de aceptación, los probadores crean pruebas basadas en las historias de usuario y sus criterios de aceptación.

En muchas situaciones, los requisitos no funcionales pueden documentarse como historias de usuario. Las técnicas de diseño de caja negra también pueden utilizarse para crear pruebas de características de calidad no funcionales.

## 3.3.4 Prueba Exploratoria

La prueba exploratoria es importante en los proyectos ágiles debido al escaso tiempo disponible para el análisis de prueba y los detalles limitados de las historias de usuario. Para conseguir los mejores resultados, la prueba exploratoria debería combinarse con otras técnicas basadas en la experiencia como parte de una estrategia de prueba reactiva, mezclada con otras estrategias de prueba como la prueba analítica basada en el riesgo, la prueba analítica basada en los requisitos, la prueba basada en modelos y la prueba adversa a la regresión.

Un contrato de prueba puede incluir la siguiente información:

* Actor: usuario previsto del sistema.
* Propósito: el tema del contrato, incluido el objetivo concreto que el actor quiere alcanzar, es decir, las condiciones de la prueba.
* Preparación: lo que hay que hacer para iniciar la ejecución de prueba.
* Prioridad: importancia relativa de este contrato, basada en la prioridad de la historia de usuario asociada o en el nivel de riesgo.
* Referencia: 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
* Notas del oráculo: cómo evaluar el producto para determinar los resultados correctos
* Variaciones: acciones y evaluaciones alternativas para complementar las ideas descritas en las actividades

La calidad de las pruebas depende de la capacidad de los probadores para plantear preguntas pertinentes sobre lo que hay que probar.

Durante la ejecución de prueba, el probador utiliza la creatividad, la intuición, el conocimiento y la competencia para encontrar posibles problemas con el producto.

Es importante que el probador documente el proceso lo máximo posible. De lo contrario, sería difícil volver atrás y ver cómo se descubrió un problema en el sistema.

La información registrada debe capturarse y/o resumirse en algún tipo de herramienta de gestión del estado de forma que facilite a los implicados la comprensión del estado actual de todas las pruebas realizadas.

# 3.4 Herramientas en Proyectos Ágiles

## 3.4.1 Herramientas de Gestión y Seguimiento de Tareas

En algunos casos, los equipos ágiles utilizan tableros físicos de historias/tareas para gestionar y hacer un seguimiento de las historias de usuario, las pruebas y otras tareas a lo largo de cada esprint. Estas herramientas sirven para lo siguiente:

* Registrar las historias y sus correspondientes tareas de desarrollo y prueba, para garantizar que no se pierda nada durante un esprint
* Capturar las estimaciones de los miembros del equipo sobre sus tareas y calcular automáticamente el esfuerzo necesario para implementar una historia, para apoyar sesiones eficientes de planificación de la iteración.
* Asociar las tareas de desarrollo y las tareas de prueba con la misma historia, para proporcionar una imagen completa del esfuerzo del equipo necesario para implementar la historia.
* Agregar las actualizaciones de los desarrolladores y probadores al estado de la tarea a medida que completan su trabajo, proporcionando automáticamente una instantánea calculada actual del estado de cada historia, la iteración y la entrega global.
* Proporcionar una representación visual del estado actual de cada historia de usuario, la iteración y la entrega, lo que permite a todos los implicados.
* Integración con herramientas de gestión de la configuración, que pueden permitir el registro de entrada de código y construcciones automatizadas con respecto a las tareas y, en algunos casos, actualizaciones de estado automatizadas de las tareas.

## 3.4.2 Herramientas de Comunicación e Intercambio de Información

Además del correo electrónico, los documentos y la comunicación verbal, los equipos ágiles suelen utilizar tres tipos de herramientas adicionales para apoyar la comunicación y el intercambio de información: las wikis, la mensajería instantánea y el escritorio compartido.

Las wikis permiten a los equipos construir y compartir una base de conocimientos en línea sobre diversos aspectos del proyecto, como los siguientes:

* Diagramas de las prestaciones del producto, discusiones sobre las prestaciones, diagramas de los prototipos, fotos de las discusiones en la pizarra, y otra información.
* Herramientas y/o técnicas de desarrollo y prueba que otros miembros del equipo consideran útiles.
* Métricas, gráficos y paneles de control sobre el estado del producto, lo que resulta especialmente útil cuando el wiki está integrado con otras herramientas, como el servidor de construcción y el sistema de gestión de tareas, ya que la herramienta puede actualizar el estado del producto automáticamente.
* Conversaciones entre los miembros del equipo, similares a la mensajería instantánea y al correo electrónico, pero de forma compartida con todos los miembros del equipo

Las herramientas de mensajería instantánea, teleconferencia de audio y videochat ofrecen las siguientes ventajas:

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

Las herramientas para compartir y capturar el escritorio proporcionan los siguientes beneficios:

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

## 3.4.3 Herramientas de Construcción y Distribución de Software

Como se expuso anteriormente en este programa de estudio, la construcción diaria y el despliegue de software es una práctica clave en los equipos ágiles. Esto requiere el uso de herramientas de integración continua y de distribución de la construcción.

## 3.4.4 Herramientas de Gestión de la Configuración

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

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

## 3.4.5 Herramientas de Diseño de Pruebas, Implementación y Ejecución

Algunas herramientas son útiles para los probadores ágiles en puntos específicos del proceso de prueba de software. Aunque la mayoría de estas herramientas no son nuevas o específicas para proyectos ágiles, proporcionan importantes capacidades dado el cambio que se produce en los proyectos ágiles.

* Herramientas de diseño de pruebas: El uso de herramientas como los mapas mentales se ha hecho muy popular para diseñar y definir rápidamente las pruebas de una nueva prestación.
* Herramientas de gestión de casos de prueba: El tipo de herramientas de gestión de casos de prueba utilizadas en un entorno ágil pueden formar parte de la gestión del ciclo de vida de la aplicación de todo el equipo o de la herramienta de gestión de tareas.
* Herramientas de preparación y generación de datos de prueba: Las herramientas que generan datos para poblar la base de datos de una aplicación son muy beneficiosas cuando se necesitan muchos datos y combinaciones de datos para probar la aplicación
* Herramientas de carga de datos de prueba: Una vez generados los datos para las pruebas, hay que cargarlos en la aplicación. La introducción manual de datos suele requerir mucho tiempo y es propensa a errores, pero existen herramientas de carga de datos que hacen que el proceso sea fiable y eficiente.
* Herramientas de ejecución de pruebas automatizadas: Existen herramientas de ejecución de prueba que están más alineadas con las pruebas ágiles. Estas herramientas permiten a los probadores y al personal de negocio expresar el comportamiento esperado del sistema en tablas o en lenguaje natural mediante palabras clave.
* Herramientas de prueba exploratoria: Las herramientas que capturan y registran las actividades realizadas en una aplicación durante una sesión de prueba exploratoria son beneficiosas para el probador y el desarrollador, ya que registran las acciones realizadas.

### 3.4.6 Informática en la Nube y Herramientas de Virtualización

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