Unidad 1
Desarrollo ágil de software

1. Fundamentos del desarrollo ágil de software

Un probador en un proyecto ágil no trabajará igual que uno que trabaje en un proyecto tradicional. Los probadores deben entender los valores y principios que sustentan los proyectos ágiles, y cómo los probadores forman parte de un enfoque de equipo completo junto con los desarrolladores y los representantes de negocio. Los miembros de un proyecto ágil se comunican entre sí de forma temprana y con frecuencia, algo que ayuda a eliminar defectos lo antes posible y a desarrollar un producto de calidad.

1.1 Desarrollo ágil de software y el manifiesto ágil

En 2001, un grupo de personas que representaban las metodologías de desarrollo de software, establecieron una serie de valores y principios comunes que pasó a conocerse como el Manifiesto por el Desarrollo Ágil de Software o el Manifiesto Ágil.

El Manifiesto Ágil recoge cuatro declaraciones 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.

El Manifiesto Ágil establece que si bien los conceptos a la derecha tienen valor, los de la izquierda tienen un valor mayor.

Individuos e Interacciones
El desarrollo ágil está muy centrado en las personas. Los equipos de personas construyen software, y la forma en la que los equipos pueden trabajar con la máxima efectividad es a través de la comunicación e interacción continua, más que basándose en herramientas o procesos.

Software que funciona
Desde el punto de vista del cliente, un software que funciona es mucho más útil y valioso que una documentación excesivamente detallada, y ofrece la oportunidad de ofrecer

feedback rápido al equipo de desarrollo. Además, dado que el software que funciona, aunque sea con funcionalidad reducida, está disponible mucho antes en el ciclo de vida de desarrollo, el desarrollo ágil puede ofrecer una importante ventaja en términos de time to market. El desarrollo Ágil es, por lo tanto, especialmente útil en entornos de negocio que cambian con rapidez donde los problemas y/o soluciones no están claros o donde el negocio desea innovar en nuevas áreas.

Colaboración del cliente

Los clientes a menudo encuentran grandes dificultades para especificar el sistema que necesitan. Al colaborar directamente con el cliente se mejora la probabilidad de entender exactamente qué necesita el cliente. Si bien tener contratos con los clientes puede ser importante, probablemente trabajar en colaboración estrecha y regular con ellos contribuya más al éxito del proyecto.

Respondiendo al cambio

El cambio es inevitable en los proyectos de software. El entorno en el que opera el negocio, la legislación, la actividad de los competidores, los avances tecnológicos y otros factores pueden afectar significativamente al proyecto y a sus objetivos. El proceso de desarrollo debe tener en cuenta estos factores. Como tal, tener flexibilidad en las prácticas de trabajo para responder ante el cambio es más importante que limitarse a cumplir estrictamente un plan.

Los valores clave del Manifiesto Ágil se resumen 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 que funciona frecuentemente, entre dos semanas y dos meses, preferentemente el 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 desarrollan 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 que no es necesario realizar, 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.

Las distintas metodologías ágiles ofrecen prácticas prescriptivas para poner en práctica estos valores y principios.

1.2 Enfoque de equipo completo

El enfoque de equipo completo significa implicar a todo el mundo con el conocimiento y las habilidades necesarios para garantizar el éxito del proyecto. El equipo incluye representantes del cliente y otras partes interesadas del negocio que determinan las prestaciones del producto. El equipo debería ser relativamente pequeño; se ha observado que la mayoría de equipos de éxito tienen entre tres y nueve miembros. Idealmente, todo el equipo comparte el mismo espacio de trabajo, dado que la co-ubicación facilita la comunicación y la interacción. El enfoque del equipo completo está soportado a través de reuniones diarias de pie en las que participan todos los miembros del equipo y donde se comunica el avance del trabajo y se destacan todos los impedimentos para el mismo. El enfoque de equipo completo promueve una dinámica de equipo más efectiva y eficiente.

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

- Mejorar la comunicación y colaboración dentro del equipo
- Permitir el aprovechamiento de las distintas habilidades dentro del equipo en beneficio del proyecto
- Hacer que la calidad sea responsabilidad de todos

Todo el equipo es responsable de la calidad en los proyectos ágiles. La esencia del enfoque de equipo completo reside en los probadores, desarrolladores y representantes de negocio que trabajan juntos en cada etapa del proceso de desarrollo. Los probadores trabajaran en estrecha colaboración tanto con desarrolladores como con representantes de negocio para garantizar que se alcanzan los niveles de calidad deseados. Esto incluye dar soporte y colaborar con los representantes de negocio para ayudarles a crear pruebas de aceptación adecuadas, trabajar con los desarrolladores para establecer conjuntamente la estrategia de pruebas y decidir sobre los enfoques de automatización de las pruebas. Los probadores pueden así transferir y extender el conocimiento de pruebas a otros miembros del equipo e influenciar en el desarrollo del producto.

Todo el equipo participa en todas las consultas o reuniones relativas a la presentación, el análisis o la estimación de las prestaciones del producto. El concepto de implicar a los probadores, desarrolladores y representantes de negocio en todas las discusiones sobre prestaciones se conoce como “El poder de tres” [Crispin08].

1.3 Feedback temprano y frecuente

Los proyectos ágiles tienen iteraciones cortas que permiten al equipo del proyecto recibir feedback temprano y de manera continua sobre la calidad del producto durante todo el ciclo de vida del desarrollo. Una forma de ofrecer feedback rápido es mediante la integración continua.

Cuando se utilizan enfoques de desarrollo secuencial, el cliente a menudo no ve el producto hasta que el proyecto está casi acabado. Llegado ese punto, suele ser demasiado tarde para que el equipo de desarrollo pueda abordar de manera efectiva los problemas que el cliente pueda tener. Al recibir feedback frecuente del cliente a medida que el proyecto avanza, los equipos ágiles pueden incorporar la mayoría de los nuevos cambios al proceso de desarrollo del producto. El feedback temprano y frecuente ayuda al equipo a concentrarse en las prestaciones que tienen mayor valor de negocio, o mayor riesgo asociado, y estas se entregan al cliente en primer lugar. También ayuda a gestionar el equipo mejor, ya que la capacidad del equipo es transparente para todos. Por ejemplo, ¿cuánto trabajo podemos hacer en un sprint o en una iteración? ¿Qué puede ayudarnos a ir más rápido? ¿Qué nos impide hacerlo?

Las ventajas del feedback temprano y frecuente incluyen:

- Evitar malentendidos de requisitos, que posiblemente no se hubieran detectado hasta más avanzado el ciclo de desarrollo cuando su reparación sería más cara.
- Aclarar requisitos de las necesidades del cliente, haciendo que estén disponibles cuanto antes para su uso por parte del cliente. De esta forma, el producto refleja mejor lo que quiere el cliente.
- Descubrir (mediante la integración continua), aislar y resolver problemas de calidad de forma temprana.
- Ofrecer información al equipo ágil sobre su productividad y capacidad para realizar entregas.
- Promover un impulso consistente dentro del proyecto.

2. Aspectos de enfoques ágiles

Hay una serie de enfoques ágiles que se están usando en las organizaciones. La práctica común en la mayoría de las organizaciones ágiles incluye la creación de historias de usuario colaborativas, retrospectivas, la integración continua y la planificación de cada iteración y de la entrega. En esta subsección se describen algunos de los enfoques ágiles.

2.1 Enfoques de desarrollo ágil de software

Hay varios enfoques de desarrollo ágil de software, cada uno de los cuales implementa los valores y principios del Manifiesto Ágil de forma distinta. En este plan de estudios, consideramos tres ejemplos de enfoques ágiles: Programación extrema (XP), Scrum, y Kanban.

Programación extrema

La Programación extrema (XP), originalmente introducida por Kent Beck [Beck04], es un enfoque de desarrollo de software descrito por determinados valores, principios y prácticas de desarrollo.

XP promueve cinco valores para guiar el desarrollo: la comunicación, la simplicidad, el feedback, la valentía y el respeto.

XP describe una serie de principios como directrices adicionales: humanidad, economía, beneficio mutuo, auto-similitud, mejora, diversidad, reflexión, flujo, oportunidad, redundancia, el fallo, calidad, pequeños pasos y la responsabilidad aceptada.

XP describe trece prácticas principales: sentarse juntos, el equipo completo, el espacio de trabajo informativo, el trabajo enérgico, la programación en parejas, las historias, el ciclo semanal, el ciclo trimestral, holgura, la construcción de software en diez minutos, la integración continua, crear pruebas antes de programar y el diseño incremental.

Muchos de los enfoques de desarrollo ágil de software utilizados actualmente están influenciados por XP y sus valores y principios. Por ejemplo, los equipos ágiles que siguen Scrum a menudo incorporan prácticas de XP.

Scrum

Scrum es un marco de trabajo de gestión ágil que contiene los siguientes instrumentos constitutivos y prácticas [Schwaber01]:

- Sprint: Scrum divide un proyecto en iteraciones (llamadas sprints) de longitud fija (normalmente de dos a cuatro semanas).
- Incremento de producto: Cada sprint acaba siendo un producto potencialmente entregable (denominado incremento).

- Backlog del producto: El propietario del producto gestiona una lista priorizada de los elementos planificados del producto (denominada backlog del producto). El backlog del producto evoluciona de sprint a sprint (lo que se conoce como refinamiento del backlog).
- Backlog del sprint: Al inicio de cada sprint, el equipo Scrum selecciona una serie de elementos de la máxima prioridad (denominado el backlog del sprint) a partir del backlog del producto. Debido a que el equipo Scrum selecciona los requisitos de la lista es un pull-principle (pedir) en lugar de un push-principle (recibir).
- Definición de "Hecho" (Terminado): Para asegurarnos de que existe un producto potencialmente entregable al final de cada sprint, el equipo Scrum debate y define los criterios de finalización del sprint. El debate profundiza en el entendimiento del equipo sobre los elementos del backlog del sprint y los requisitos del producto.
- Timeboxing (Límite de tiempo): Solo aquellas tareas, requisitos, o prestaciones que el equipo espera completar dentro del sprint forman parte del backlog del sprint. Si el equipo de desarrollo no puede completar una tarea dentro de un sprint, las prestaciones del producto asociadas se eliminan del sprint y la tarea pasa de nuevo al backlog del producto. El concepto de timeboxing no solo se refiere a tareas, sino también a otras situaciones (por ejemplo, hacer cumplir la hora de inicio y fin en las reuniones).
- Transparencia: El equipo de desarrollo reporta y actualiza el estado del sprint a diario en una reunión llamada el scrum diario. Esto hace que el contenido y el avance del sprint actual, incluidos los resultados de las pruebas sean visibles para el equipo, la dirección y todas las partes interesadas. Por ejemplo, el equipo de desarrollo puede mostrar el estado del sprint en una pizarra.

Scrum define tres funciones:

- Scrum Master: garantiza que las prácticas y reglas de scrum se implementan y se siguen, y resuelve cualquier infracción, problema de recursos o demás obstáculos que puedan impedir al equipo seguir las prácticas y normas. Esta persona no es el jefe del equipo, sino un coach.
- Propietario del producto: representa al cliente y genera, mantiene y prioriza el backlog del producto. Esta persona no es el jefe del equipo.
- Equipo de desarrollo: desarrolla y prueba el producto. El equipo se organiza a sí mismo: No hay un jefe, por lo que es el equipo el que toma las decisiones. El equipo también es multidisciplinario.

Scrum (al contrario que en XP) no establece técnicas específicas de desarrollo de software (por ejemplo, crear pruebas antes de programar). Además, Scrum no ofrece orientación sobre cómo deben hacerse las pruebas en un proyecto Scrum.

Kanban [Anderson13] 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 [Linz14]:

- Tablero Kanban: La cadena de valor a gestionar se visualiza a través de un tablero Kanban. Cada columna muestra un estado, que es una serie de actividades relacionadas, por ejemplo, desarrollo o pruebas. Los elementos a producir o las tareas a procesar están representados por tickets que se mueven de izquierda a derecha en el tablero a través de los estados.

- Límite de tareas en curso: La cantidad de tareas activas en paralelo está estrictamente limitada. Se controla a través del número máximo de tickets permitidos para un estado y/o para todo el tablero. Cuando haya capacidad libre en una estación, el trabajador sacará un ticket del estado anterior.

- Tiempo de entrega: Kanban se utiliza para optimizar el flujo continuo de tareas minimizando el tiempo de entrega (promedio) necesario para todo el flujo de valor.

Kanban tiene ciertas similitudes con Scrum. En ambos marcos de trabajo, visualizar las tareas activas (por ejemplo, en una pizarra pública) proporciona transparencia en cuanto al contenido y al progreso de las tareas. Las tareas que aún no han sido programadas se quedan en espera en un backlog de producto y pasan al tablero Kanban en cuanto hay espacio libre (capacidad productiva) disponible.

Las iteraciones o los sprints son opcionales en Kanban. El proceso Kanban permite llevar a cabo entregables ítem por ítem en lugar de como parte de una entrega. El timeboxing como mecanismo de sincronización, es opcional, al contrario que en Scrum, que sincroniza todas las tareas dentro de un sprint.

2.2 Creación de historias de usuario colaborativas

A menudo, una de las principales causas del fracaso de un proyecto son las especificaciones deficientes. Pueden surgir problemas de especificación a causa de la falta de conocimiento por parte del cliente de sus verdaderas necesidades, la ausencia de una visión global del sistema, unas funcionalidades contradictorias o redundantes, así como otros fallos de comunicación. En el desarrollo ágil, las historias de usuario se escriben para establecer los requisitos desde las perspectivas de los desarrolladores, los probadores y los representantes de negocio. En el desarrollo secuencial, esta visión compartida de una funcionalidad se consigue a través de revisiones formales una vez los requisitos están escritos; en el desarrollo

ágil, esta visión compartida se consigue a través de frecuentes revisiones informales a medida que se escriben los requisitos.

Las historias de usuario deben abordar características tanto funcionales como no funcionales. Cada historia incluye criterios de aceptación para estas características. Estos criterios deberían definirse conjuntamente entre los representantes de negocio, los desarrolladores y los probadores. Proporcionan a los desarrolladores y probadores una visión ampliada de la funcionalidad que los representantes de negocio validarán. Un equipo ágil considera una tarea acabada cuando se cumplen una serie de criterios de aceptación

Para la autoría colectiva de la historia de usuario puede recurrirse a técnicas como la lluvia de ideas y los mapas mentales. El probador puede utilizar la técnica INVEST [INVEST]:

- Independiente
- Negociable
- Valiosa
- Estimable
- Pequeña
- Testeable

Según el concepto de las 3 Cs [Jeffries00], una historia de usuario es la conjunción de tres elementos:

- Tarjeta (Card, en inglés): La tarjeta es el medio físico donde se describe una historia de usuario. Identifica el requisito, su criticidad, la duración esperada del desarrollo y pruebas, así como los criterios de aceptación para dicha historia. La descripción tiene que ser exacta, ya que se utilizará en el backlog del producto.

- Conversación: La conversación explica cómo se utilizará el software. La conversación puede ser documentada o verbal. Los probadores, que tienen un punto de vista distinto del de los desarrolladores y representantes de negocio [ISTQB_FL_SYL], hacen una valiosa aportación al intercambio de ideas, opiniones y experiencias. La conversación se inicia durante la fase de planificación de la entrega y continúa cuando se programa la historia.

- Confirmación: Los criterios de aceptación, tratados en la conversación, se utilizan para confirmar que la historia está hecha. Estos criterios de aceptación pueden abarcar múltiples historias de usuario. Deberían utilizarse pruebas tanto positivas como negativas para cubrir los criterios. Durante la confirmación, varios participantes hacen el papel de probador. Entre ellos puede haber tanto desarrolladores como especialistas centrados en el rendimiento, la seguridad, la interoperabilidad y otras características de la calidad. Para confirmar una historia como "Hecha", deben probarse los criterios de aceptación definidos y demostrar que se cumplen.

Los equipos ágiles varían en términos de cómo documentan las historias de usuario. Independientemente del enfoque adoptado para documentar las historias de usuario, la documentación debe ser concisa, suficiente y necesaria.

2.3 Retrospectivas (Reuniones retrospectivas)

En el desarrollo ágil, una retrospectiva es una reunión realizada al final de cada iteración para discutir qué ha tenido éxito, qué puede mejorarse y cómo incorporar las mejoras y repetir los éxitos en iteraciones futuras. Las retrospectivas cubren temas como el proceso, las personas, las organizaciones, las relaciones y las herramientas. Las retrospectivas periódicas, cuando se llevan a cabo las actividades de seguimiento adecuadas, son críticas para la auto-organización y la mejora continua del desarrollo y las pruebas.

Las retrospectivas pueden dar lugar a decisiones de mejora de las pruebas centradas en la eficacia de las pruebas, la productividad de las pruebas, la calidad de los casos de prueba y la satisfacción del equipo. También pueden abordar la facilidad de probar las aplicaciones, las historias de usuario, las funcionalidades o las interfaces de sistema. El análisis de la causa raíz de los defectos puede guiar las mejoras de las pruebas y del desarrollo. En general, los equipos deben centrarse exclusivamente en algunas cuestiones por iteración, y por lo tanto deben implementar solo unas cuantas mejoras por iteración. Esto permite la mejora continua a un ritmo sostenido.

Los tiempos y la organización de la retrospectiva dependen del método ágil que se siga. Los representantes de negocio y el equipo asisten a las retrospectivas en calidad de participantes, mientras que el facilitador organiza y dirige la reunión. En algunos casos, los equipos pueden invitar a otros participantes a la reunión.

Los probadores deben desempeñar un papel importante en las retrospectivas. Los probadores forman parte del equipo y aportan su perspectiva [ISTQB_FL_SYL], Sección 1.5. Las pruebas se realizan en cada sprint y contribuyen de forma vital al éxito. Todos los miembros del equipo, tanto probadores como no probadores, pueden aportar información sobre actividades de prueba y del resto de actividades.

Las retrospectivas deben celebrarse en un entorno profesional caracterizado por la confianza mutua. Los atributos de una retrospectiva de éxito son los mismos que los de cualquier otra revisión, según lo previsto en la Sección 3.2 del Plan de estudios de Nivel Básico [ISTQB_FL_SYL].

2.4 Integración continua

La entrega de un incremento de producto requiere un software confiable, que funcione e integrado al final de cada sprint. La integración continua encara este reto fusionando todos los cambios realizados al software e integrando todos los componentes desarrollados de forma periódica, al menos una vez al día. La gestión de la configuración, la compilación, la construcción de software, el despliegue y las pruebas se unen en un único proceso automatizado y repetible. Dado que los desarrolladores integran su trabajo de forma constante, construyen de forma constante, y prueban de forma constante, se detectan más rápido los defectos en el código.

Una vez que los desarrolladores han codificado, depurado y subido el código en un repositorio de código compartido, el proceso de integración continua consta de las siguientes actividades automatizadas:

- Análisis estático del código: ejecuta un análisis estático del código y reporta los resultados
- Compilación: compila y enlaza el código, generando los ficheros ejecutables
- Pruebas unitarias: ejecuta las pruebas unitarias, comprueba la cobertura de código y reporta los resultados de las pruebas.
- Despliegue: instala en un entorno de pruebas el software que se ha construido
- Prueba de integración: ejecuta las pruebas de integración y reporta los resultados.
- Información (Cuadro de mando): informa del estado de todas estas actividades en una ubicación pública y visible o comunica el estado al equipo por correo electrónico.

A diario se realiza un proceso automatizado de compilación (build) y pruebas para detectar errores de integración de forma temprana y rápida. La integración continua permite a los probadores ágiles ejecutar pruebas automatizadas, en algunos casos como parte del propio proceso de integración continua, y proporcionar feedback rápido al equipo sobre la calidad del código. Los resultados de estas pruebas están visibles para todos los miembros del equipo, especialmente cuando hay informes automatizados integrados en el proceso. Las pruebas automatizadas de regresión pueden ser continuas durante toda la iteración. Unas buenas pruebas automatizadas de regresión cubren el máximo posible de funcionalidad, incluyendo historias de usuario entregadas en iteraciones anteriores. Una buena cobertura en las pruebas automatizadas de regresión ayuda a la construcción (y pruebas) de sistemas integrados grandes. Cuando se automatizan las pruebas de regresión, los probadores ágiles quedan liberados para concentrar sus pruebas manuales en nuevas funcionalidades, cambios implementados y pruebas de confirmación de arreglos de defectos.

Además de las pruebas automatizadas, las organizaciones que usan integración continua normalmente utilizan herramientas de compilación (build) para implementar un control de calidad continuo. Además de ejecutar pruebas unitarias y de integración, esas herramientas pueden ejecutar pruebas estáticas y dinámicas adicionales para medir y perfilar el

rendimiento, extraer y formatear documentación del código fuente, y facilitar procesos manuales de aseguramiento de la calidad. Esta aplicación continua de un control de calidad tiene por objeto mejorar la calidad del producto así como reducir el tiempo que lleva su entrega sustituyendo la práctica tradicional de aplicar el control de calidad una vez completado todo el desarrollo.

Las herramientas de compilación pueden estar vinculadas a herramientas de despliegue automático, que pueden localizar el software construido a partir de la integración continua o el servidor de compilación y desplegarlo en uno o más entornos de desarrollo, pruebas y producción. Esto supone una reducción de los errores y los retrasos asociados a la dependencia de personal especializado o programadores para instalar entregas en estos entornos.

La integración continua puede ofrecer las siguientes ventajas:

- Permite una detección temprana y un análisis más sencillo de la causa raíz de problemas de integración y cambios conflictivos
- Aporta al equipo de desarrollo feedback frecuente sobre si el código está o no funcionando.
- Preserva la versión del software sujeto a pruebas en el día de la versión en desarrollo.
- Reduce el riesgo de regresión asociado a la refactorización del código del desarrollador gracias a la rápida repetición de las pruebas del código base después de cada pequeño grupo de cambios.
- Proporciona confianza de que el trabajo de desarrollo diario tiene una base sólida.
- Hace visible el avance hacia la finalización del incremento del producto, lo que anima a los desarrolladores y probadores.
- Elimina los riesgos previstos asociados a la integración tipo big-bang.
- Ofrece disponibilidad constante de software ejecutable durante todo el sprint para pruebas, demostraciones o formación.
- Reduce las actividades de pruebas manuales repetitivas.
- Ofrece un rápido feedback sobre las decisiones adoptadas para mejorar la calidad y las pruebas.

Sin embargo, la integración continua no está exenta de riesgos y desafíos:

- Las herramientas de integración continua tienen que implantarse y mantenerse.
- El proceso de integración continua debe definirse y establecerse.
- La automatización de pruebas requiere recursos adicionales y puede ser difícil de establecer.
- Una cobertura de pruebas exhaustiva es esencial para lograr ventajas de las pruebas automatizadas.
- Los equipos a menudo confían demasiado en las pruebas unitarias y llevan a cabo muy pocas pruebas de sistema y pruebas de aceptación.

La integración continua requiere el uso de herramientas, incluyendo herramientas para pruebas, herramientas para automatizar el proceso de construcción y herramientas para el control de versiones.

2.5 Planificación de la entrega y planificación de la iteración

Según lo mencionado en el plan de estudios de Nivel Básico [ISTQB_FL_SYL], la planificación es una actividad constante, y éste también es el caso en los ciclos de vida ágiles. En los ciclos de vida ágiles, se dan dos tipos de planificación, la planificación de la entrega y la planificación de la iteración.

La planificación de la entrega prevé la entrega de un producto, a menudo unos cuantos meses antes del inicio de un proyecto. La planificación de la entrega define y redefine el backlog del producto, y puede implicar la redefinición de historias de usuario grandes en una colección de historias más pequeñas. La planificación de la entrega proporciona la base para un enfoque de pruebas y un plan de pruebas de todas las iteraciones. Las planificaciones de la entrega son de alto nivel.

En la planificación de la entrega, los representantes de negocio establecen y priorizan las historias de usuario para la entrega, en colaboración con el equipo.Tomando como base estas historias de usuario, se identifican los riesgos de proyecto y los riesgos de calidad y se realiza una estimación de esfuerzo a alto nivel.

Los probadores participan en la planificación de la entrega y aportan especial valor en las siguientes actividades:

- Definición de historias de usuario testeables, incluyendo criterios de aceptación
- Participación en los análisis de proyecto y análisis de riesgos de calidad
- Estimación del esfuerzo de pruebas asociado a las historias de usuario
- Definición de los niveles de pruebas necesarios
- Planificación de las pruebas para la entrega

Una vez realizada la planificación de la entrega, se inicia la planificación de la primera iteración. La planificación de la iteración se extiende hasta el final de una única iteración y se refiere solo al backlog de la iteración.

En la planificación de la iteración, el equipo selecciona las historias de usuario a partir del backlog de entregas priorizadas, elabora las historias de usuario, lleva a cabo un análisis de riesgos de las historias de usuario, y estima el trabajo necesario para cada historia de usuario. Si una historia de usuario es demasiado vaga y los intentos por aclararla han fracasado, el equipo puede evitar aceptarla y utilizar la siguiente historia de usuario en base a las prioridades. Los representantes de negocio deben responder a las preguntas del equipo

sobre cada historia de forma que el equipo pueda entender qué deben implementar y cómo deben probar cada historia.

El número de historias seleccionado depende de la velocidad establecida para el equipo y del tamaño estimado de las historias de usuario seleccionadas. Una vez finalizados los contenidos de la iteración, las historias de usuario se dividen en tareas, que se llevarán a cabo por los correspondientes miembros del equipo.
Los probadores participan en la planificación de la iteración y aportan especial valor en las siguientes actividades:

- Participar en el análisis de riesgos detallados de las historias de usuario
- Determinar la testabilidad de las historias de usuario
- Crear pruebas de aceptación para las historias de usuario
- Dividir las historias de usuario en tareas (particularmente las tareas de pruebas)
- Calcular el esfuerzo de prueba para todas las tareas de prueba.
- Identificar aspectos funcionales y no funcionales del sistema a probar.
- Dar soporte y participar en la automatización de pruebas en los diferentes niveles de pruebas.

Las planificaciones de la entrega pueden variar a medida que el proyecto va avanzando, incluyendo cambios en historias de usuario del backlog del producto. Estos cambios pueden surgir por factores internos o externos. Entre los factores internos se incluyen la capacidad de entrega, la velocidad y problemas técnicos. Los factores externos pueden ser el descubrimiento de nuevos mercados y oportunidades, nuevos competidores, o amenazas de negocio que pueden cambiar los objetivos y/o las fechas de entrega. Además, la planificación de las iteraciones puede cambiar durante una iteración. Por ejemplo, una historia de usuario que fue considerada durante la estimación relativamente sencilla podría resultar ser más compleja de lo esperado.

Estos cambios pueden suponer un reto para los probadores. Los probadores deben tener una visión global de la entrega para la planificación de las pruebas, y deben disponer de una base de pruebas de un oráculo de pruebas adecuados en cada iteración para el desarrollo de las pruebas según lo mencionado en la Sección 1.4 del plan de estudios de Nivel Básico [ISTQB_FL_SYL]. El probador debe disponer lo antes posible de la información necesaria, y sin embargo los cambios deben aceptarse según los principios ágiles. Este dilema requiere decisiones meditadas sobre la estrategia de pruebas y sobre la documentación de las pruebas.

La planificación de la entrega y la planificación de la iteración deben abordar la planificación de las pruebas y la planificación de las actividades de desarrollo. Algunos temas específicos relacionados con las pruebas que deben abordarse son:

- El alcance de las pruebas, la extensión de las pruebas para las áreas dentro del alcance, los objetivos de las pruebas y la justificación de estas decisiones.

- Los miembros del equipo que llevarán a cabo las actividades de pruebas.
- El entorno de pruebas y los datos de prueba necesarios, cuándo se necesitan y si se producirán adiciones o cambios en el entorno y/o en los datos de prueba antes o durante el proyecto.
- Los tiempos, las secuencias, las dependencias y los requisitos previos para las actividades de prueba funcionales y no funcionales (por ejemplo, con cuánta frecuencia se ejecutan pruebas de regresión, qué prestaciones dependen de otras prestaciones o datos de prueba, etc.), incluyendo cómo las actividades de pruebas dependen de las actividades de desarrollo.
- Los riesgos del proyecto y de calidad que deben abordarse.

Además, el esfuerzo de estimación del equipo debe tener en cuenta el tiempo y el esfuerzo necesarios para llevar a cabo las actividades de pruebas requeridas.