Probador Certificado del ISTQB®
Programa de Estudio de Nivel Básico
Extensión Ágil

Traducción realizada por el

Spanish Software Testing Qualifications Board con el apoyo del

Hispanic America Software Testing Qualifications Board

Versión ES 001.08

Basada en el Programa de Estudio

“Certified Tester, Foundation Level Extension Syllabus Agile Tester, Version 2014”

International Software Testing Qualifications Board

Nota sobre Derechos de Propiedad Intelectual.

Foundation Level Extension Agile Tester Working Group: Rex Black (Chair), Bertrand Cornanguer (Vice Chair), Gerry Coleman (Learning Objectives Lead), Debra Friedenberg (Exam Lead), Alon Linetzki (Business Outcomes y Marketing Lead), Tauhida Parveen (Editor), y Leo van der Aalst (Development Lead).

Autores: Rex Black, Anders Claesson, Gerry Coleman, Bertrand Cornanguer, Istvan Forgacs, Alon Linetzki, Tilo Linz, Leo van der Aalst, Marie Walsh, y Stephan Weber.

Revisores internos: Mette Bruhn-Pedersen, Christopher Clements, Alessandro Collino, Debra Friedenberg, Kari Kakkonen, Beata Karpinska, Sammy Kolluru, Jennifer Leger, Thomas Mueller, Tuula Pääkkönen, Meile Posthuma, Gabor Puhalla, Lloyd Roden, Marko Rytkönen, Monika Stoecklein-Olsen, Robert Treffny, Chris Van Bael, y Erik van Veenendaal; 2013-2014.

Historial de Revisiones

Versión	Fecha	Observaciones
Programa de Estudio v0.1	26JUL2013	Secciones independientes.
Programa de Estudio v0.2	16SEP2013	Se incorporan los comentarios de revisión del GT sobre la v01.
Programa de Estudio v0.3	20OCT2013	Se incorporan los comentarios de revisión del GT sobre la v02.
Programa de Estudio v0.7	16DEC2013	Se incorporan los comentarios de revisión de Alpha sobre la v03.
Programa de Estudio v0.71	20DEC2013	Actualizaciones del grupo de trabajo sobre la v07.
Programa de Estudio v0.9	30JAN2014	Versión Beta.
Programa de Estudio 2014	31MAY2014	Versión GA.
Programa de Estudio 2014	30SEP2014	Corrección de errores menores.

Tabla de Contenidos

Agradecimientos

Este documento ha sido elaborado por un equipo de International Software Testing Qualifications Board Foundation Level Working Group.

El equipo de Agile Extension agradece al equipo revisor y a los Comités Nacionales por sus sugerencias y aportaciones.

Cuando se completó el programa de estudio de nivel básico de la Extensión Ágil, el Grupo de Trabajo de la Extensión Ágil estaba compuesto por los siguientes miembros: Rex Black (Chair), Bertrand Cornanguer (Vice Chair), Gerry Coleman (Learning Objectives Lead), Debra Friedenberg (Exam Lead), Alon Linetzki (Business Outcomes and Marketing Lead), Tauhida Parveen (Editor), y Leo van der Aalst (Development Lead).

Autores: Rex Black, Anders Claesson, Gerry Coleman, Bertrand Cornanguer, Istvan Forgacs, Alon Linetzki, Tilo Linz, Leo van der Aalst, Marie Walsh, y Stephan Weber.

Revisores internos: Mette Bruhn-Pedersen, Christopher Clements, Alessandro Collino, Debra Friedenberg, Kari Kakkonen, Beata Karpinska, Sammy Kolluru, Jennifer Leger, Thomas Mueller, Tuula Pääkkönen, Meile Posthuma, Gabor Puhalla, Lloyd Roden, Marko Rytkönen, Monika Stoecklein-Olsen, Robert Treffny, Chris Van Bael, y Erik van Veenendaal.

El equipo agradece también a las siguientes personas, de los Comités Nacionales y de la comunidad de expertos en Agile, que participaron en la revisión, los comentarios y la votación del programa de estudio de la Extensión Ágil: Dani Almog, Richard Berns, Stephen Bird, Monika Bögge, Afeng Chai, Josephine Crawford, Tibor Csöndes, Huba Demeter, Arnaud Foucal, Cyril Fumery, Kobi Halperin, Inga Hansen, Hanne Hinz, Jidong Hu, Phill Isles, Shirley Itah, Martin Klonk, Kjell Lauren, Igal Levi, Rik Marselis, Johan Meivert, Armin Metzger, Peter Morgan, Ninna Morin, Ingvar Nordstrom, Chris O'Dea, Klaus Olsen, Ismo Paukamainen, Nathalie Phung, Helmut Pichler, Salvatore Reale, Stuart Reid, Hans Rombouts, Petri Säilynoja, Soile Sainio, Lars-Erik Sandberg, Dakar Shalom, Jian Shen, Marco Sogliani, Lucjan Stapp, Yaron Tsubery, Sabine Uhde, Stephanie Ulrich, Tommi Välimäki, Jurian Van de Laar, Marnix Van den Ent, António Vieira Melo, Wenye Xu, Ester Zabar, Wenqiang Zheng, Peter Zimmerer, Stevan Zivanovic, y Terry Zuo.

Este documento fue aprobado formalmente para su entrega por la Asamblea General del ISTQB® el 31 de mayo de 2014.

Notas de la Versión en Idioma Español

El Spanish Software Testing Qualifications Board (SSTQB) ha llevado a cabo la traducción del Programa de Estudio de “ISTQB® Certified Tester, Foundation Level Extension, Agile Tester” versión de 2014. Este Programa de Estudio se denomina, en idioma español, “Probador Certificado del ISTQB®” de “Nivel Básico – Extensión Ágil” versión 2014.

El equipo de traducción y revisión para este programa de estudio es el siguiente (por orden alfabético):

Responsable de la traducción: Gustavo Márquez Sosa (España)

Revisora: Luisa Morales Gómez Tejedor (España)

El Comité Ejecutivo del SSTQB agradece especialmente las aportaciones de los revisores.

En una siguiente versión se podrán incorporar aportaciones adicionales. El SSTQB considera conveniente mantener abierta la posibilidad de realizar cambios en el “Programa de Estudio”.

Madrid, 24 de febrero de 2019

Traducciones Específicas, Definiciones, Acrónimos, Abreviaturas y Notas

En este capítulo se presentarán traducciones específicas, definiciones, acrónimos, abreviaturas y notas de términos y conceptos que no forman parte del Glosario de Términos del ISTQB y tampoco están incluidos en su traducción al idioma español o castellano. Se ha incorporado este apartado para facilitar la lectura y comprensión de esta "Programa de Estudio".

Es conveniente observar que las traducciones de los distintos términos, sean o no del glosario, se han realizado con el objetivo de evitar conflictos en sus traducciones y siguiendo las normas de traducción del SSTQB. Algunos conflictos surgen en el momento de traducir un nuevo término o un nuevo programa de estudio.

Traducciones específicas		
Español	**Inglés**	**Observaciones**
competencia	skill	Este término también se puede traducir como: • capacidad.
tablero de tareas	task board	Este término también se puede traducir como: • panel de tareas.
entrega	release	Este término también se puede traducir como: • publicar. • publicación. • lanzar. • lanzamiento.
entrenamiento	coaching	Este término también se puede traducir como: • entrenamiento profesional.
trabajo en pareja	pairing	Este término también se puede traducir como: • emparejamiento
lista de trabajo acumulado del producto	product backlog	Este término también se puede traducir como: • pila del producto • cartera del producto
Concepto 3C	3C concept	No hay observaciones.
desarrollo guiado por prueba de aceptación	acceptance test-driven development	No hay observaciones.
prueba de aceptación	acceptance test	No hay observaciones.
tablero de tareas ágil	Agile task board	No hay observaciones.
refinamiento del trabajo acumulado	backlog refinement	No hay observaciones.

Traducciones específicas		
Español	**Inglés**	**Observaciones**
desarrollo guiado por el comportamiento	behavior-driven development	No hay observaciones.
gráfico de quemado	burndown chart	No hay observaciones.
implicado del negocio	business stakeholder	No hay observaciones.
ubicación común	co-location	No hay observaciones.
retroalimentación contínua	continuous feedback	No hay observaciones.
integración continua	continuous integration	No hay observaciones.
colaboración con el cliente	customer collaboration	No hay observaciones.
reunión de pie diaria	daily stand-up meeting	No hay observaciones.
herramienta de generación de datos	data generator tools	No hay observaciones.
taxonomía de defectos	defect taxonomy	No hay observaciones.
definición de hecho	definition of done	No hay observaciones.
definición de preparado	definition of ready	No hay observaciones.
ensayo	dry run	No hay observaciones.
épica	epic	No hay observaciones.
técnica basada en expertos	expert-based technique	No hay observaciones.
programación extrema	Extreme Programming	No hay observaciones.
prestación	feature	No hay observaciones.
retroalimentación	feedback	No hay observaciones.
dado/cuando/entonces	given/when/then	No hay observaciones.
incremento	increment	No hay observaciones.
INVEST	INVEST	No hay observaciones.
planificación de la iteración	iteration planning	No hay observaciones.
Kanban	Kanban	No hay observaciones.
tablero Kanban	Kanban board	No hay observaciones.
prueba en pareja	pair testing	No hay observaciones.
póquer de planificación	planning poker	No hay observaciones.
poder de tres	power of three	No hay observaciones.
mejora del proceso	process improvement	No hay observaciones.
propietario de producto	product owner	No hay observaciones.
producto de trabajo de proyecto	project work product	No hay observaciones.
análisis del riesgo de calidad	quality risk analysis	No hay observaciones.
planificación de la entrega	release planning	No hay observaciones.
retrospectiva	retrospective	No hay observaciones.

Traducciones específicas		
Español	**Inglés**	**Observaciones**
análisis de la causa raíz	root cause analysis	No hay observaciones.
Scrum	Scrum	No hay observaciones.
Scrum Master	Scrum Master	No hay observaciones.
prueba de seguridad	security testing	No hay observaciones.
equipo autoorganizado	self-organizing teams	No hay observaciones.
esprint	sprint	No hay observaciones.
trabajo acumulado del esprint	sprint backlog	No hay observaciones.
reunión de pie	stand-up meeting	No hay observaciones.
tarjeta de historia	story card	No hay observaciones.
puntos de historia	story points	No hay observaciones.
susceptibilidad a desbordamientos de la memoria intermedia	susceptibility to buffer overflows	No hay observaciones.
desarrollo sostenible	sustainable development	No hay observaciones.
deuda técnica	technical debt	No hay observaciones.
herramienta de preparación de datos de prueba	test data preparation tools	No hay observaciones.
programar probando primero	test first programming	No hay observaciones.
pirámide de prueba	test pyramid	No hay observaciones.
modelo de cuadrantes de prueba	testing quadrant model	No hay observaciones.
cuadrantes de prueba	testing quadrants	No hay observaciones.
tiempo de comercialización	time to market	No hay observaciones.
acotamiento del tiempo	timeboxing	No hay observaciones.
transparencia	transparency	No hay observaciones.
doce principios	twelve principles	No hay observaciones.
prueba de usabilidad	usability testing	No hay observaciones.
velocidad	velocity	No hay observaciones.
control de versiones	version control	No hay observaciones.
enfoque de equipo completo	whole-team approach	No hay observaciones.
software funcionando	working software	No hay observaciones.
	XP	No hay observaciones.

Tabla 1 - Traducciones Específicas

Definiciones, Acrónimos y Abreviaturas			
Tipo de Término	**Idioma del Acrónimo**	**Término**	**Descripción**
ABREVIATURA	Español	N/A	**N**o **A**plica
ACRÓNIMO	Inglés	ATDD	**A**cceptance **T**est **D**riven **D**evelopment
ACRÓNIMO	Inglés	BDD	**B**ehavior **D**riven **D**evelopment
ACRÓNIMO	Inglés	DoD	**D**efinition **o**f **D**one
ACRÓNIMO	Inglés	QA	**Q**uality **A**ssurance
ACRÓNIMO	Inglés	TDD	**T**est **D**riven **D**evelopment

Tabla 2 – Definiciones, Acrónimos y Abreviaturas

0 Introducción

0.1 Objetivo de este Programa de Estudio

Este programa de estudio constituye la base para la formación como Probador Certificado del ISTQB® de Nivel Básico - Extensión Ágil. El ISTQB proporciona este programa de estudio en los siguientes términos:

1. A los comités miembro, para traducir a su idioma local y para acreditar a los proveedores de formación. Los comités miembros pueden adaptar el programa de estudio a sus necesidades lingüísticas particulares y añadir referencias para adaptarlo a sus publicaciones locales.
2. A los organismos de certificación, para elaborar las preguntas del examen en su lengua local adaptadas a los objetivos de aprendizaje de este programa de estudio.
3. A los proveedores de formación, para desarrollar material didáctico y determinar los métodos de enseñanza adecuados.
4. A los candidatos a la certificación, para que preparen el examen de certificación (ya sea como parte de un curso de formación o de forma independiente).
5. A la comunidad internacional de ingeniería de software y sistemas, para avanzar en la profesión de prueba del software y sistemas, y como fuente de libros y artículos.

El ISTQB puede permitir que otras entidades utilicen este programa de estudio para otros fines, siempre y cuando soliciten y obtengan permiso previo y por escrito por parte del ISTQB.

0.2 El Nivel Básico de Probador Certificado en la Prueba de Software

El documento ISTQB® Foundation Level Agile Tester Overview es un documento adicional que incluye la siguiente información:

- Resultados del negocio para el programa de estudio.
- Resumen del programa de estudio.
- Relaciones entre los programas de estudio.
- Descripción de los niveles cognitivos (niveles K)
- Apéndices.

Desarrollo Ágil de Software

Duración: 150 minutos	
Palabras clave	
automatización de la prueba	test automation
base de prueba	test basis

ciclo de vida del software	software lifecycle
desarrollo ágil de software	agile software development
desarrollo guiado por pruebas	test-driven development
historia de usuario	user story
manifiesto ágil	agile manifesto
modelo de desarrollo incremental	incremental development model
modelo de desarrollo iterativo	iterative development model
oráculo de prueba	test oracle

Objetivos de Aprendizaje para el Capítulo "Desarrollo Ágil de Software":		
1.1. Fundamentos de Desarrollo Ágil de Software		
NB-AT-1.1.1	(K1)	Recordar el concepto básico del desarrollo ágil de software basado en el Manifiesto Ágil.
NB-AT-1.1.2	(K2)	Comprender las ventajas del enfoque de equipo completo.
NB-AT-1.1.3	(K2)	Comprender los beneficios de la retroalimentación temprana y frecuente.
1.2. Características de los Enfoques Ágiles		
NB-AT-1.2.1	(K2)	Recordar los enfoques de desarrollo ágil de software
NB-AT-1.2.2	(K3)	Redactar historias de usuario, que puedan ser objeto de prueba, en colaboración con desarrolladores y representantes de negocio.
NB-AT-1.2.3	(K2)	Comprender cómo se pueden utilizar las retrospectivas como mecanismo para la mejora del proceso en proyectos ágiles.
NB-AT-1.2.4	(K2)	Comprender el uso y el objetivo de la integración continua.
NB-AT-1.2.5	(K2)	Conocer las diferencias entre la planificación de la iteración y la planificación de la entrega, y cómo un probador añade valor en cada una de estas actividades

1.1 Fundamentos de Desarrollo Ágil de Software

Un probador en un proyecto ágil trabajará de forma diferente a uno que trabaje en un proyecto tradicional. Los probadores deben comprender los valores y principios en los que se basan los proyectos ágiles, y cómo los probadores son parte integrante de un enfoque de equipo completo junto con los desarrolladores y representantes de negocio. Los miembros de un proyecto ágil se comunican entre sí desde el principio y con frecuencia, lo que ayuda a eliminar los defectos desde el principio y a desarrollar un producto de calidad.

1.1.1 El Desarrollo Ágil de Software y el Manifiesto Ágil

En 2001, un grupo de personas, que representaban las metodologías de desarrollo ágil de software más utilizadas, se pusieron de acuerdo en un conjunto de valores y principios comunes que se conocieron como el Manifiesto para el Desarrollo Ágil de Software o el Manifiesto Ágil [Agilemanifesto]. 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

El Manifiesto Ágil sostiene que, aunque los conceptos de la derecha tienen valor, los de la izquierda tienen más valor.

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. Además, dado que el software funcionando, aunque con una funcionalidad reducida, está disponible mucho antes en el ciclo de vida de desarrollo, el desarrollo ágil puede conferir una importante ventaja en el tiempo de comercialización. 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. Aunque

tener contratos con los clientes puede ser importante, trabajar en estrecha colaboración con ellos de forma regular probablemente aportará un mayor éxito al proyecto.

Respuesta ante el 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 tener una gran influencia en el proyecto y sus objetivos. 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.

Las diferentes metodologías ágiles ofrecen prácticas prescriptivas para llevar a la práctica estos valores y principios.

1.1.2 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 equipo debe ser relativamente pequeño; se han observado equipos de éxito con tan sólo tres personas y hasta nueve. Lo ideal es que todo el equipo comparta el mismo espacio de trabajo, ya que la ubicación común facilita mucho la comunicación y la interacción. El enfoque de equipo completo se apoya en las reuniones de pie diarias (véase la sección 2.2.1) 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 todos.

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. Los probadores trabajarán estrechamente con los desarrolladores y los representantes de negocio para garantizar que se alcanzan los niveles de calidad deseados. Esto incluye apoyar y colaborar con los representantes de negocio para ayudarles a crear pruebas de aceptación adecuadas, trabajar con los desarrolladores para acordar la estrategia de pruebas y decidir los enfoques de automatización de la prueba. Los probadores pueden así transferir y ampliar los conocimientos sobre pruebas a otros miembros del equipo e influir en el desarrollo del producto.

Todo el equipo participa en las consultas o reuniones en las que se presentan, analizan o estiman las prestaciones del producto. El concepto de involucrar a probadores, desarrolladores y representantes de negocio en todas las discusiones sobre las prestaciones se conoce como el poder de tres [Crispin08].

1.1.3 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. Una forma de proporcionar una retroalimentación rápida es la integración continua (véase el apartado 1.2.4).

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. La retroalimentación temprana y frecuente ayuda al equipo a centrarse en las prestaciones con mayor valor de negocio, o riesgo asociado, y éstas se entregan primero al cliente. También ayuda a gestionar mejor el equipo, ya que su capacidad es transparente para todos. Por ejemplo, ¿cuánto trabajo podemos hacer en un esprint o iteración? ¿Qué podría ayudarnos a ir más rápido? ¿Qué nos impide hacerlo?

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 los mismos 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

Hay una serie de enfoques ágiles que utilizan las organizaciones. Las prácticas comunes en la mayoría de las organizaciones ágiles incluyen la creación de historias de usuario en colaboración, las retrospectivas, la integración continua y la planificación de cada iteración, así como de la entrega global. Esta subsección describe algunos 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 [Beck04], 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.

Muchos de los enfoques de desarrollo ágil de software que se utilizan hoy en día están influidos por XP y sus valores y principios. Por ejemplo, los equipos ágiles que siguen Scrum suelen incorporar prácticas de XP.

Scrum

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

- 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 (denominada trabajo acumulado del producto). El trabajo acumulado del producto evoluciona de un esprint a otro (lo que se denomina refinamiento del trabajo acumulado).
- Trabajo Acumulado del Esprint: Al comienzo de cada Sprint, el equipo Scrum selecciona un conjunto de elementos de máxima prioridad (llamado trabajo acumulado del esprint) a partir del trabajo acumulado del producto. Dado que el equipo Scrum, y no el propietario del producto, selecciona los elementos que se van a realizar dentro del esprint, la selección se refiere al principio de extracción en lugar del principio de inserción.
- 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. La discusión refuerza la comprensión del equipo de los elementos del trabajo acumulado y de los requisitos del producto.
- Acotamiento del Tiempo: Sólo las tareas, requisitos o prestaciones que el equipo espera terminar dentro del esprint forman parte del trabajo acumulado del esprint. Si el equipo de desarrollo no puede terminar una tarea dentro de un esprint, las prestaciones del producto asociadas se eliminan del esprint y la tarea se traslada de nuevo al trabajo acumulado del producto. El acotamiento del tiempo no sólo se aplica a las tareas, sino también a otras situaciones (por ejemplo, el cumplimiento de las horas de inicio y finalización de las reuniones).
- Transparencia: El equipo de desarrollo informa y actualiza diariamente el estado del esprint en una reunión denominada scrum diario. Esto hace que el contenido y el progreso del esprint 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 esprint en una pizarra.

Scrum define tres roles:

- Scrum Master: 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 (véase la sección 2.3.2 y la sección 3.1.4).

Kanban

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 que se gestiona se visualiza mediante un tablero Kanban. Cada columna muestra una estación, que es un conjunto de actividades relacionadas, por ejemplo, desarrollo o pruebas. Los elementos que se van a producir o las tareas que se van a procesar se simbolizan mediante tickets que se desplazan de izquierda a derecha por el tablero a través de las estaciones.
- 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. Cada vez que una estación tiene capacidad libre, el trabajador extrae un ticket de la estación predecesora.
- 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.

Kanban presenta algunas similitudes con Scrum. En ambos marcos, la visualización de las tareas activas (por ejemplo, en una pizarra pública) aporta transparencia al contenido y al progreso de las tareas. Las tareas que aún no están programadas esperan en una lista de trabajo acumulado y se trasladan al tablero Kanban en cuanto hay un nuevo espacio (capacidad de producción) disponible.

Las iteraciones o esprints son opcionales en Kanban. El proceso Kanban permite liberar sus entregables elemento por elemento, en lugar de hacerlo como parte de una entrega. El acotamiento del tiempo como mecanismo de sincronización, por lo tanto, es opcional, a diferencia de Scrum, que sincroniza todas las tareas dentro de un esprint.

1.2.2 Creación Colaborativa de Historias de Usuario

Las especificaciones deficientes suelen ser una de las principales razones del fracaso de los proyectos. Los problemas de especificación pueden resultar de la falta de conocimiento de los usuarios sobre sus verdaderas necesidades, de la ausencia de una visión global del sistema, de prestaciones redundantes o contradictorias y de otros errores de comunicación. En el desarrollo ágil, las historias de usuario se escriben para capturar los requisitos desde las perspectivas de los desarrolladores, probadores y representantes de negocio. En el desarrollo secuencial, esta visión compartida de una prestación se logra a través de revisiones formales después de escribir los requisitos; en el desarrollo ágil, esta visión compartida se logra a través de frecuentes revisiones informales mientras se escriben los requisitos.

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. Proporcionan a los desarrolladores y probadores una visión ampliada de la prestación que los representantes de negocio validarán. Un equipo ágil considera que una tarea está terminada cuando se han satisfecho un conjunto de criterios de aceptación.

Normalmente, la perspectiva única del probador mejorará la historia de usuario identificando los detalles que faltan o los requisitos no funcionales. Un probador puede contribuir haciendo preguntas abiertas a los representantes de negocio sobre la historia de usuario, proponiendo formas de probar la historia de usuario y confirmando los 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 [INVEST]:

Español	**Inglés**
Independiente	**I**ndependent
Negociable	**N**egotiable
Valioso	**V**aluable
Estimable	**E**stimable
Pequeño	**S**mall
Capacidad de Ser Probado	**T**estable

Según el concepto 3C [Jeffries00], 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. Los probadores, al tener un punto de vista diferente al de los desarrolladores y representantes de negocio [ISTQB_FL_SYL], aportan una valiosa entrada al intercambio de ideas, opiniones y experiencias. La conversación comienza durante la fase de planificación de la entrega y continúa cuando se programa la historia.
- 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. Durante la confirmación, varios participantes desempeñan el papel de probador. Entre ellos pueden estar los desarrolladores, así como especialistas centrados en el rendimiento, la seguridad, la interoperabilidad y otras características de calidad. Para confirmar que una historia está hecha, se deben probar los criterios de aceptación definidos y demostrar que se cumplen.

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

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 reuniones retrospectivas realizadas con regularidad, cuando se realizan las actividades de seguimiento adecuadas, son fundamentales para la autoorganización y la mejora continua del desarrollo y la prueba.

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. También pueden abordar la capacidad de ser probada de las aplicaciones, las historias de usuario, las prestaciones o las interfaces del sistema. El análisis de la causa raíz de los defectos puede impulsar mejoras en la prueba y el desarrollo. En general, los equipos deberían implementar sólo unas pocas mejoras por iteración. Esto permite una mejora continua a un ritmo sostenido.

El momento y la organización de la retrospectiva dependen del método ágil concreto que se siga. Los representantes de negocio y el equipo asisten a cada retrospectiva como 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 deberían desempeñar un papel importante en las retrospectivas. Los probadores forman parte del equipo y aportan su perspectiva única [ISTQB_FL_SYL], sección 1.5. La prueba se produce en cada esprint y contribuye de forma vital al éxito. 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.

Las retrospectivas deben producirse 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, tal y como se discute en el Programa de Estudio del Nivel Básico [ISTQB_FL_SYL], sección 3.2.

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. La gestión de la configuración, la compilación, la construcción del software, el despliegue y las pruebas se integran en un proceso único, automatizado y repetible. Como los desarrolladores integran su trabajo constantemente, construyen constantemente y prueban constantemente, los defectos en el código se detectan más rápidamente.

A continuación de la codificación, depuración y comprobación del código por parte de los desarrolladores en un repositorio de código fuente compartido, 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.

Un proceso automatizado de compilación y prueba tiene lugar a diario y detecta los errores de integración de forma temprana y rápida. 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. Estos resultados de prueba

son visibles para todos los miembros del equipo, especialmente cuando los informes automatizados se integran en el proceso. Las pruebas de regresión automatizadas pueden ser continuas durante toda la iteración. Unas buenas pruebas de regresión automatizadas cubren toda la funcionalidad posible, incluidas las historias de usuario entregadas en las iteraciones anteriores. Una buena cobertura en las pruebas de regresión automatizadas ayuda a la construcción (y prueba) de grandes sistemas integrados. Cuando las pruebas de regresión están automatizadas, los probadores ágiles quedan liberados para concentrar sus pruebas manuales en las nuevas prestaciones, los cambios implementados y las pruebas de confirmación de las correcciones de defectos.

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. Esta aplicación continua del control de la calidad tiene como objetivo mejorar la calidad del producto, así como reducir el tiempo de entrega del mismo, sustituyendo la práctica tradicional de aplicar el control de la calidad después de completar todo el desarrollo.

Las herramientas de construcción pueden vincularse a herramientas de despliegue automático, que pueden obtener la construcción adecuada del servidor de integración continua o de construcción y desplegarla en uno o más entornos de desarrollo, prueba, puesta en escena o incluso de producción. Esto reduce los errores y retrasos asociados a la dependencia de personal especializado o programadores para instalar las entregas en estos entornos.

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.

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

- Las herramientas de integración continua deben ser introducidas y mantenidas.

- Hay que definir y establecer el proceso de integración continua.
- La automatización de la prueba requiere recursos adicionales y puede ser compleja de establecer.
- La cobertura profunda de la prueba es esencial para lograr las ventajas de las pruebas automatizadas.
- Los equipos a veces confían demasiado en las pruebas unitarias y realizan muy pocas pruebas de aceptación y de sistema.

La integración continua requiere el uso de herramientas, incluidas las de prueba, las de automatización del proceso de construcción y las de control de versiones.

1.2.5 Planificación de Entregas e Iteraciones

Como se menciona en el programa de estudio del nivel básico [ISTQB_FL_SYL], 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, y puede implicar el refinamiento de las historias de usuario más grandes en una colección de historias más pequeñas. La planificación de entrega proporciona la base para un enfoque de prueba y un plan de prueba que abarca todas las iteraciones. Los planes de 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 (véase el apartado 1.2.2). A partir de estas historias de usuario, se identifican los riesgos de proyecto y de calidad y se realiza una estimación del esfuerzo a alto nivel (véase la sección 3.2).

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.

Una vez realizada la planificación de la entrega, comienza la planificación de la iteración para la primera iteración. La planificación de la iteración se adelanta al final de una sola iteración y se ocupa de la lista de trabajo acumulado de la iteración.

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. Si una historia de usuario es demasiado vaga y los intentos de aclararla han fallado, el equipo puede rechazarla y utilizar la siguiente historia de usuario en función de la prioridad. Los representantes de negocio deben responder a las preguntas del equipo sobre cada historia para que éste pueda entender qué debe implementar y cómo probar cada historia.

El número de historias seleccionadas se basa en la velocidad establecida del equipo y en el tamaño estimado de las historias de usuario seleccionadas. Una vez finalizado el contenido de la iteración, las historias de usuario se dividen en tareas, que serán llevadas a cabo por los miembros del equipo correspondientes.

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. Los factores internos incluyen la capacidad de entrega, la velocidad y las cuestiones técnicas. Los factores externos incluyen el descubrimiento de nuevos mercados y oportunidades, nuevos competidores o amenazas de negocio que pueden cambiar los objetivos de entrega y/o las fechas previstas. Además, los planes de iteración pueden cambiar durante una iteración. Por ejemplo, una historia de usuario concreta que se consideraba relativamente sencilla durante la estimación puede resultar más compleja de lo esperado.

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 y un oráculo de prueba en cada iteración a efectos de desarrollo de la prueba, tal como se discute en el programa de estudio del nivel básico [ISTQB_FL_SYL], sección 1.4.

La información necesaria debe estar disponible para el probador con antelación y, sin embargo, hay que aceptar los cambios de acuerdo con los principios ágiles. Este dilema requiere decisiones cuidadosas sobre las estrategias de prueba y la documentación de prueba. Para saber más sobre los retos de las pruebas ágiles, consulte [Black09], capítulo 12.

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, cuándo se necesitan y si se producirán adiciones o cambios en el entorno de prueba y/o en los datos antes o durante el proyecto.
- El momento, la secuencia, las dependencias y los prerrequisitos para las actividades de prueba funcionales y no funcionales (por ejemplo, la frecuencia con la que se ejecutan las pruebas de regresión, qué prestaciones dependen de otras prestaciones o de los datos de prueba, etc.),

incluido el modo en que las actividades de prueba se relacionan con las actividades de desarrollo y dependen de ellas.

- Los riesgos de proyecto y de calidad que deben abordarse (véase la sección 3.2.1).

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

Duración: 105 minutos	
Palabras clave	
prueba de verificación de la construcción	build verification test
elemento de la configuración	configuration item
gestión de la configuración	configuration management

Objetivos de Aprendizaje para el Capítulo "Principios, Prácticas y Procesos Fundamentales de Prueba Ágil":		
2.1 Las Diferencias entre Probar en Enfoques Tradicionales y Ágiles		
NB-AT-2.1.1	(K2)	Describir las diferencias entre las actividades de prueba en proyectos ágiles y en proyectos no ágiles.
NB-AT-2.1.2	(K2)	Describir cómo se integran las actividades de desarrollo y prueba en los proyectos ágiles.
NB-AT-2.1.3	(K2)	Describir el papel de la prueba independiente en los proyectos ágiles.
2.2 Situación de la Prueba en Proyectos Ágiles		
NB-AT-2.2.1	(K2)	Describir las herramientas y técnicas utilizadas para comunicar el estado de la prueba en un proyecto ágil, incluyendo el avance de la prueba y la calidad del producto.
NB-AT-2.2.2	(K2)	Describir el proceso de evolución de las pruebas a través de múltiples iteraciones y explicar por qué la automatización de la prueba es importante para gestionar el riesgo de regresión en los proyectos ágiles.
2.3 Rol y Competencias de un Probador en un Equipo Ágil		
NB-AT-2.3.1	(K2)	Comprender las competencias (personas, dominio y pruebas) de un probador en un equipo ágil.
NB-AT-2.3.2	(K2)	Comprender el rol de un probador en un equipo ágil.

2.1 Las Diferencias entre Probar en Enfoques Tradicionales y Ágiles

Como se describe en el programa de estudio del nivel básico [ISTQB_FL_SYL] y en [Black09], 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 (por ejemplo, el secuencial como el modelo-V o el iterativo como RUP) y los ciclos de vida ágiles para poder trabajar con eficacia y eficiencia. Los modelos ágiles difieren en cuanto a la forma en que se integran las actividades de prueba y desarrollo, los productos de trabajo del proyecto, los nombres, los criterios de entrada y salida utilizados para los distintos niveles de prueba, el uso de herramientas y la forma en que se pueden utilizar eficazmente la prueba independiente.

Los probadores deben recordar que las organizaciones varían considerablemente en su aplicación de los ciclos de vida. La desviación de los ideales de los ciclos de vida ágiles (véase la sección 1.1) puede representar una personalización y adaptación inteligente de las prácticas. La capacidad de adaptación al contexto de un proyecto determinado, incluidas las prácticas de desarrollo de software realmente seguidas, es un factor clave de éxito para los probadores.

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. Al principio del proyecto, hay un periodo de planificación de la entrega. A éste le sigue una secuencia de iteraciones. Al principio de cada iteración, hay un periodo de planificación de la iteración. Una vez establecido el alcance de la iteración, se desarrollan las historias de usuario seleccionadas, se integran en el sistema y se prueban. Estas iteraciones son muy dinámicas, con actividades de desarrollo, integración y prueba que tienen lugar a lo largo de cada iteración, y con un considerable paralelismo y solapamiento. Las actividades de prueba tienen lugar a lo largo de la iteración, no como una actividad final.

Los probadores, los desarrolladores y los implicados del negocio tienen todos 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.

En algunos casos, las iteraciones de consolidación o estabilización se producen periódicamente para resolver cualquier defecto persistente y otras formas de deuda técnica. Sin embargo, la buena práctica es que ninguna prestación se considere hecha hasta que se haya integrado y probado con el sistema [Goucher09]. Otra buena práctica es abordar los defectos que quedan de la iteración anterior al principio de la siguiente, como parte de la lista de trabajo acumulado para esa iteración (lo que se conoce como "arreglar primero los bugs"). Sin embargo, algunos se quejan de que esta práctica da lugar a una situación en la que se desconoce el trabajo total a realizar en la iteración y será más difícil estimar cuándo se podrán realizar las prestaciones restantes. Al final de la secuencia de iteraciones, puede haber un conjunto de actividades de entrega para que el software esté listo para ser entregado, si bien en algunos casos la entrega se produce al final de cada iteración.

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. Sin embargo, los riesgos de calidad específicos asociados a cada iteración se identifican y evalúan en la planificación de la iteración. Este análisis del riesgo puede influir en la secuencia de desarrollo, así como en la prioridad y la profundidad de la prueba de las prestaciones. También influye en la estimación del esfuerzo de prueba necesario para cada prestación (véase la sección 3.2).

En algunas prácticas ágiles (por ejemplo, la Programación Extrema), 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. El trabajo en pareja puede ser difícil cuando el equipo de prueba está distribuido, pero los procesos y las herramientas pueden ayudar a permitir el trabajo en pareja distribuido. Para más información sobre el trabajo distribuido, véase [ISTQB_ALTM_SYL], sección 2.8.

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. Esto promueve un sentido de propiedad colectiva de la calidad del producto.

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. Debido al gran uso de la automatización de pruebas, un mayor porcentaje de las pruebas manuales en los proyectos ágiles tiende a realizarse mediante técnicas basadas en la experiencia y en los defectos, como los ataques de software, la prueba exploratoria y la predicción de errores (véase [ISTQB_ALTA_SYL], secciones 3.3 y 3.4, y [ISTQB_FL_SYL], sección 4.5). Mientras que los desarrolladores se centrarán en la creación de pruebas unitarias, los probadores deberían centrarse en la creación de pruebas automatizadas de integración, de sistema y de integración de sistemas. Esto hace que los equipos ágiles tiendan a favorecer a los probadores con una sólida formación técnica y de automatización de la prueba.

Un principio ágil fundamental es que el cambio puede producirse a lo largo del proyecto. Por lo tanto, en los proyectos ágiles se favorece la documentación ligera del producto de trabajo. Los cambios en las prestaciones existentes tienen implicaciones en la prueba, especialmente en la prueba de regresión. El uso de pruebas automatizadas es una forma de gestionar la cantidad de esfuerzo de prueba asociado al cambio. Sin embargo, es importante que el ritmo de los cambios no supere la capacidad del equipo del proyecto para hacer frente a los riesgos asociados a esos cambios.

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 (por ejemplo, documentación de usuario).
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 (por ejemplo, el código) o que evalúan piezas individuales de código (por ejemplo, pruebas unitarias automatizadas).
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 (por ejemplo, pruebas manuales y automatizadas), o que presentan los resultados de las pruebas (por ejemplo, tableros de control de las pruebas, tal y como se comenta en la sección 2.2.1)

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. Este estímulo para reducir la documentación sólo se aplica a la documentación que no aporta valor al cliente. En un proyecto ágil de éxito, se alcanza un equilibrio entre el aumento de la eficiencia mediante la reducción de la documentación y la provisión de documentación suficiente para apoyar las actividades de negocio, pruebas, desarrollo y mantenimiento. El equipo debe tomar una decisión durante la planificación de la versión sobre qué productos de trabajo se necesitan y qué nivel de documentación de los productos de trabajo se necesita.

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. Una historia de usuario debe definir una prestación lo suficientemente pequeña como para ser completada en una sola iteración. Las colecciones más grandes de características relacionadas, o una colección de subcaracterísticas que conforman una única característica compleja, pueden denominarse "épicas". Las épicas pueden incluir historias de usuario para diferentes equipos de desarrollo. Por ejemplo, una historia de usuario puede describir lo que se necesita a nivel de la API (middleware) mientras que otra historia describe lo que se necesita a nivel de la interfaz de usuario (aplicación). Estas colecciones pueden desarrollarse a lo largo de una serie de esprints. Cada épica y sus historias de usuario deben tener criterios de aceptación asociados.

Los productos de trabajo típicos de los desarrolladores en los proyectos ágiles incluyen el código. Los desarrolladores ágiles también suelen crear pruebas unitarias automatizadas. Estas pruebas pueden crearse tras el desarrollo del código. Sin embargo, en algunos casos, los desarrolladores crean pruebas de forma incremental, antes de que se escriba cada fragmento de código, con el fin de proporcionar una forma de verificar, una vez escrita esa porción de código, si funciona como se esperaba. Aunque este enfoque se denomina "probar primero" o "desarrollo guiado por pruebas", en realidad las pruebas son más una forma de especificaciones de diseño de bajo nivel ejecutables que pruebas [Beck02].

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. Estos documentos se capturan de la forma más ligera posible, lo que también suele ocurrir con estos documentos en los ciclos de vida tradicionales. Los probadores también producirán métricas de pruebas a partir de informes de defectos y registros de resultados de pruebas, y de nuevo se hace hincapié en un enfoque ligero.

En algunas implementaciones ágiles, especialmente en proyectos y productos regulados, de seguridad física, distribuidos o muy complejos, se requiere una mayor formalización de estos productos de trabajo. Por ejemplo, algunos equipos transforman las historias de usuario y los criterios de aceptación en especificaciones de requisitos más formales. Se pueden preparar informes de trazabilidad vertical y horizontal para cumplir con auditores, reglamentos y otros requisitos.

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. En algunos modelos iterativos, esta regla no se aplica. Los niveles de prueba se solapan. La especificación de requisitos, la especificación de diseño y las actividades de desarrollo pueden solaparse con los niveles de prueba.

En algunos ciclos de vida ágiles, el solapamiento se produce porque los cambios en los requisitos, el diseño y el código pueden producirse en cualquier momento de una iteración. Aunque Scrum, en teoría, no permite cambios en las historias de usuario después de la planificación de la iteración, en la práctica a veces se producen dichos cambios. 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, y consisten en probar los criterios de aceptación de la historia de usuario.
  - Prueba de validación de la prestación, que suele ser manual y puede involucrar a los desarrolladores, probadores y a los implicados del negocio que trabajan en colaboración para determinar si la prestación es apta para su uso, para mejorar la visibilidad del avance realizado y para recibir una retroalimentación real de los implicados de 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. Esto puede implicar la ejecución de pruebas funcionales, así como de pruebas no funcionales de rendimiento, fiabilidad, usabilidad y otros tipos de pruebas relevantes.

Los equipos ágiles pueden emplear varias formas de pruebas de aceptación (utilizando el término tal y como se explica en el programa de estudios de nivel básico [ISTQB_FL_SYL]). Las pruebas alfa internas y las pruebas beta externas pueden tener lugar, ya sea al final de cada iteración, tras la finalización de cada iteración o tras una serie de iteraciones. Las pruebas de aceptación de usuario, las pruebas de aceptación operativa, las pruebas de aceptación regulatoria y las pruebas de aceptación de contrato también pueden producirse, ya sea al cierre de cada iteración, después de la compleción de cada iteración o después de una serie de iteraciones.

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. Los desarrolladores registran continuamente el código y las pruebas unitarias en un sistema de gestión de la configuración, utilizando marcos de construcción y prueba automatizados. Estos marcos permiten la integración continua del nuevo software con el sistema, con el análisis estático y las pruebas unitarias ejecutadas repetidamente a medida que se registra la entrada del nuevo software [Kubaczkowski].

Estas pruebas automatizadas también pueden incluir pruebas funcionales a nivel de integración y de sistema. Estas pruebas funcionales automatizadas pueden crearse utilizando arneses de prueba funcionales, herramientas de prueba funcional de interfaz de usuario de código abierto o herramientas comerciales, y pueden integrarse con las pruebas automatizadas que se ejecutan como parte del marco de integración continua. En algunos casos, debido a la duración de las pruebas funcionales, éstas se

separan de las pruebas unitarias y se ejecutan con menos frecuencia. Por ejemplo, las pruebas unitarias pueden ejecutarse cada vez que se registre la entrada de un nuevo software, mientras que las pruebas funcionales más largas se ejecutan sólo cada pocos días.

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. Esto requiere una inversión en informes de prueba en tiempo real para proporcionar una buena visibilidad de los resultados de la prueba. Este enfoque ayuda a reducir los costosos e ineficientes ciclos de "construcción-instalación-fallo-reconstrucción-reinstalación" que pueden darse en muchos proyectos tradicionales, ya que los cambios que interrumpen la construcción o hacen que el software falle al instalarse se detectan rápidamente.

Las herramientas de construcción y pruebas automatizadas ayudan a gestionar el riesgo de regresión asociado a los frecuentes cambios que suelen producirse en los proyectos ágiles. Sin embargo, confiar demasiado en las pruebas unitarias automatizadas para gestionar estos riesgos puede ser un problema, ya que las pruebas unitarias suelen tener una efectividad limitada en la detección de defectos [Jones11]. También son necesarias las pruebas automatizadas a nivel de integración y de sistema.

2.1.5 Opciones de Organización para Pruebas Independientes

Como se comenta en el programa de estudio de nivel básico [ISTQB_FL_SYL], 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. Sin embargo, dada la posición de esos probadores dentro del equipo, se corre el riesgo de perder la independencia y la evaluación objetiva.

Otros equipos ágiles mantienen equipos de pruebas totalmente independientes y separados, y asignan probadores a demanda durante los últimos días de cada esprint. Esto puede preservar la independencia, y estos probadores pueden proporcionar una evaluación objetiva e imparcial del software. Sin embargo, las limitaciones de tiempo, la falta de comprensión de las nuevas prestaciones del producto y los problemas de relación con los implicados del negocio y los desarrolladores suelen provocar problemas con este enfoque.

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. Además, el equipo de prueba independiente puede contar con probadores especializados ajenos a los equipos ágiles para trabajar en actividades a largo plazo y/o independientes de la iteración, como el desarrollo de herramientas de prueba automatizadas, la realización de pruebas no funcionales, la creación y el apoyo de entornos de prueba y datos, y la realización de niveles de prueba que podrían no encajar bien en un esprint (por ejemplo, pruebas de integración de sistemas).

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. Además, los cambios pueden afectar

a las prestaciones existentes de iteraciones anteriores. Por lo tanto, las pruebas manuales y automatizadas deben actualizarse para afrontar con eficacia el riesgo de regresión.

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 probadores de los equipos ágiles utilizan varios métodos para registrar el avance y el estado de las pruebas, incluidos los resultados de la automatización de la prueba, la progresión de las tareas e historias de prueba en el tablero de tareas ágil y los gráficos de quemado que muestran el progreso del equipo. A continuación, se pueden comunicar al resto del equipo utilizando medios como los paneles de control wiki y los correos electrónicos tipo panel de control, así como verbalmente durante las reuniones de pie. Los equipos ágiles pueden utilizar herramientas que generan automáticamente informes de estado basados en los resultados de las pruebas y el progreso de las tareas, que a su vez actualizan los paneles de control y los correos electrónicos de estilo wiki. Este método de comunicación también recoge métricas del proceso de prueba, que pueden utilizarse en la mejora del proceso. Comunicar el estado de las pruebas de forma tan automatizada también libera el tiempo de los probadores para concentrarse en el diseño y la ejecución de más casos de prueba.

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 [Crispin08] 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, incluido el estado de la prueba, los equipos pueden 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 (véase el apartado 1.2.5) se plasman en el tablero de tareas, a menudo utilizando tarjetas coordinadas por colores para determinar el tipo de tarea. Durante la iteración, el avance se gestiona mediante el movimiento de estas tareas a través del tablero de tareas en columnas como por hacer, trabajo en curso, verificar y hecho. Los equipos ágiles pueden utilizar herramientas para mantener sus tarjetas de historias y tableros de tareas ágil, que pueden automatizar los paneles de control y las actualizaciones de estado.

Las tareas de prueba del tablero de tareas están relacionadas con los criterios de aceptación definidos para las historias de usuario. A medida que los guiones de automatización de la prueba, las pruebas manuales y las pruebas exploratorias de una tarea de prueba alcanzan un estado de aprobación, la tarea pasa a la columna de realizado del tablero de tareas. 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. Si alguna tarea (incluidas las tareas de prueba) no avanza o lo hace con demasiada lentitud, el equipo revisa y aborda cualquier problema que pueda estar bloqueando el avance de esas tareas.

La reunión de pie diaria incluye a todos los miembros del equipo ágil, incluidos los probadores. En esta reunión, comunican su estado actual. La agenda para cada miembro es [Agile Alliance Guide]:

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

Cualquier problema que pueda bloquear el avance de la prueba se comunica durante las reuniones de pie diarias, para que todo el equipo esté al tanto de los problemas y pueda resolverlos en consecuencia.

Para mejorar la calidad general del producto, muchos equipos ágiles realizan encuestas de satisfacción del cliente para obtener retroalimentación sobre si el producto cumple las expectativas del cliente. Los equipos pueden utilizar otras métricas similares a las recopiladas en las metodologías de desarrollo tradicionales, como las tasas de paso/fallo de pruebas, las tasas de descubrimiento de defectos, los resultados de las pruebas de confirmación y regresión, la densidad de defectos, los defectos encontrados y corregidos, la cobertura de requisitos, la cobertura de riesgos, la cobertura de código y el batido de código para mejorar la calidad del producto.

Como en cualquier ciclo de vida, las métricas recopiladas e informadas deben ser relevantes y ayudar a la toma de decisiones. Las métricas no deben utilizarse para premiar, castigar o aislar a ningún miembro del equipo.

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. Además de probar los cambios de código realizados en la iteración actual, los probadores también tienen que verificar que no se ha introducido ninguna regresión en las prestaciones que se desarrollaron y probaron en iteraciones anteriores. 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. También es fundamental que todos los activos de prueba, como las pruebas automatizadas, los casos de prueba manuales, los datos de prueba y otros artefactos de prueba, se mantengan actualizados en cada iteración. Se recomienda especialmente que todos los activos de prueba se mantengan en una herramienta de gestión de la configuración para permitir el control de las versiones, garantizar la facilidad de acceso de todos los miembros del equipo y apoyar la realización de cambios según sea necesario debido a los cambios de funcionalidad, al tiempo que se conserva la información histórica de los activos de prueba.

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, y para retirar los casos de prueba que ya no son relevantes. Las pruebas redactadas en iteraciones anteriores para verificar prestaciones específicas pueden tener poco valor en iteraciones posteriores debido a los cambios en las prestaciones o a las nuevas prestaciones que alteran el modo en que se comportan esas prestaciones anteriores.

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. Esto permite que las pruebas de regresión automatizadas reduzcan el riesgo de regresión con menos esfuerzo del que requerirían las pruebas de regresión manuales. Este menor esfuerzo en las pruebas de regresión libera a los probadores para que puedan probar más a fondo las nuevas prestaciones y funciones de la iteración en curso.

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. La definición de cómo el equipo diseña, redacta y almacena los casos de prueba debe producirse durante la planificación de la entrega. Las buenas prácticas para el diseño de pruebas y su implementación deben adoptarse pronto y aplicarse de forma coherente. 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 automatizadas bien redactadas proporcionan un documento vivo de la funcionalidad del sistema [Crispin08]. Al comprobar las pruebas automatizadas y sus correspondientes resultados en el sistema de gestión de la configuración, alineado con el versionado de las construcciones del producto, los equipos ágiles pueden revisar la funcionalidad probada y los resultados de las pruebas de cualquier construcción en un momento dado.

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. Para reducir las interrupciones en la construcción, que pueden ralentizar el avance de todo el equipo, no debe registrarse la entrada de código a menos que se pasen todas las pruebas unitarias automatizadas. Los resultados de las pruebas unitarias automatizadas proporcionan una retroalimentación inmediata sobre la calidad del código y de la construcción, pero no sobre la calidad del producto.

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, pero generalmente no se ejecutan con cada registro de entrada del código, ya que tardan más en ejecutarse que las pruebas unitarias automatizadas y podrían ralentizar los registros de entrada del código. Los resultados de las pruebas de aceptación automatizadas proporcionan retroalimentación sobre la calidad del producto con respecto a la retroalimentación desde la última construcción, pero no proporcionan el estado de la calidad general del producto.

Se pueden ejecutar pruebas automatizadas de forma continua contra el sistema. Un subconjunto inicial de pruebas automatizadas para cubrir la funcionalidad y los puntos de integración críticos del sistema debería crearse inmediatamente después de que se despliegue una nueva construcción en el entorno de prueba. 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.

Las pruebas automatizadas contenidas en el conjunto de prueba de regresión se ejecutan generalmente como parte de la construcción principal diaria en el entorno de integración continua, y de nuevo cuando se despliega una nueva construcción en el entorno de prueba. En cuanto falla una prueba de regresión automatizada, el equipo se detiene e investiga las razones del fallo de la prueba. La prueba puede haber fallado debido a cambios funcionales legítimos en la iteración actual, en cuyo caso puede ser necesario actualizar la prueba y/o la historia de usuario para reflejar los nuevos criterios de aceptación. También

puede ser necesario retirar la prueba si se ha construido otra para cubrir los cambios. Sin embargo, si la prueba falló debido a un defecto, es una buena práctica que el equipo corrija el defecto antes de avanzar con las nuevas prestaciones.

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 [ISTQB_FL_SYL]. Además de estas competencias, un probador en un equipo ágil debe ser competente en la automatización de la prueba, el desarrollo guiado por pruebas, la prueba de aceptación, caja blanca, caja negra y la prueba basada en la experiencia.

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

- Ser positivos y estar orientados a las soluciones con los miembros del equipo y los implicados.
- Mostrar un pensamiento crítico, orientado a la calidad y escéptico sobre el producto.
- Recabar información de los implicados de forma activa (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.

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

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. Además de las actividades descritas en otras partes de este programa de estudio, estas actividades incluyen:

- Comprender, implementar y actualizar la estrategia de prueba.
- Medir e informar de la cobertura de la prueba en todas las dimensiones de cobertura aplicables.
- Asegurar el uso adecuado de las herramientas de prueba.
- Configurar, utilizar y gestionar los entornos de prueba y los datos de prueba.
- Informar de los defectos y trabajar con el equipo para resolverlos.
- Entrenar a otros miembros del equipo en los aspectos relevantes de la prueba.
- Asegurar que se programen las tareas de prueba adecuadas durante la planificación de la entrega y la iteración.
- Colaborar activamente con los desarrolladores y los implicados del negocio para aclarar los requisitos, especialmente en términos de capacidad de ser probado, consistencia y completitud.
- Participar de forma proactiva en las retrospectivas del equipo, sugiriendo e implementando mejoras.

Dentro de un equipo ágil, cada miembro del equipo es responsable de la calidad del producto y desempeña un papel en la realización de tareas relacionadas con la prueba.

Las organizaciones ágiles pueden encontrar algunos riesgos organizativos relacionados con la prueba:

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

Para mitigar estos riesgos, las organizaciones pueden considerar las variantes para preservar la independencia que se comentan en la sección 2.1.5.

3 Métodos, Técnicas y Herramientas de Prueba Ágiles

Duración: 480 minutos	
Palabras clave	
automatización de la ejecución de prueba	test execution automation
contrato de prueba	test charter
criterios de aceptación	acceptance criteria
desarrollo guiado por pruebas	test-driven development
enfoque de prueba	test approach
estimación de la prueba	test estimation
estrategia de prueba	test strategy
marco de trabajo de pruebas unitarias	unit test framework
prueba de regresión	regression testing
prueba de rendimiento	performance testing
prueba exploratoria	exploratory testing
riesgo de calidad	quality risk
riesgo de producto	product risk

Objetivos de Aprendizaje para el Capítulo "Métodos, Técnicas y Herramientas de Prueba Ágiles":		
3.1 Métodos de Prueba Ágil		
NB-AT-3.1.1	(K1)	Recordar los conceptos de desarrollo guiado por pruebas, desarrollo guiado por prueba de aceptación y desarrollo guiado por el comportamiento.
NB-AT-3.1.1	(K1)	Recordar los conceptos de la pirámide de prueba.
NB-AT-3.1.1	(K2)	Resumir los cuadrantes de prueba y sus relaciones con los niveles de prueba y los tipos de prueba.
NB-AT-3.1.1	(K3)	Practicar el rol de probador en un equipo Scrum para un proyecto ágil determinado

Objetivos de Aprendizaje para el Capítulo "Métodos, Técnicas y Herramientas de Prueba Ágiles":		
3.2 Evaluación de los Riesgos de Calidad y Estimación del Esfuerzo de Prueba		
NB-AT-3.2.1	(K3)	Evaluar los riesgos de calidad en un proyecto ágil.
NB-AT-3.2.2	(K3)	Estimar el esfuerzo de prueba basado en el contenido de la iteración y los riesgos de calidad.
3.3 Técnicas en Proyectos Ágiles		
NB-AT-3.3.1	(K3)	Interpretar información relevante para apoyar las actividades de prueba.
NB-AT-3.3.2	(K2)	Explicar a los implicados del negocio cómo definir criterios de aceptación que puedan ser objeto de prueba.
NB-AT-3.3.3	(K3)	Dada una historia de usuario, escribir casos de prueba de aceptación en un desarrollo guiado por pruebas.
NB-AT-3.3.4	(K3)	Para el comportamiento funcional y no funcional, escribir casos de prueba utilizando técnicas de diseño de prueba de caja negra basadas en historias de usuario dadas.
NB-AT-3.3.5	(K3)	Realizar pruebas exploratorias para apoyar la prueba de un proyecto ágil.
3.4 Herramientas en Proyectos Ágiles		
NB-AT-3.4.1	(K1)	Recordar las diferentes herramientas disponibles para los probadores de acuerdo a su propósito y a las actividades en proyectos 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, concentrándose en la prevención, detección y eliminación temprana de defectos, y asegurándose de que se ejecutan los tipos de prueba adecuados en el momento oportuno y como parte del nivel de prueba correcto. Los profesionales ágiles pretenden introducir estas prácticas en una fase temprana. Los probadores de los proyectos ágiles desempeñan un papel fundamental a la hora de guiar el uso de estas prácticas de prueba a lo largo del ciclo de vida.

3.1.1 Desarrollo Guiado por Pruebas, Desarrollo Guiado por Pruebas de Aceptación y Desarrollo Guiado por el Comportamiento

El desarrollo guiado por pruebas, el desarrollo guiado por pruebas de aceptación y el desarrollo guiado por el comportamiento son tres técnicas complementarias que se utilizan en 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 (TDD, por sus siglas en inglés) se utiliza para desarrollar código guiado por casos de prueba automatizados. El proceso para el desarrollo guiado por pruebas es:

- Se añadir una prueba que capture el concepto del programador sobre el funcionamiento deseado de un pequeño fragmento de código.
- Se ejecuta la prueba, que debería fallar ya que el código no existe.
- Se escribe el código y se ejecuta la prueba en un bucle cerrado hasta que la prueba pase.
- Se refactoriza el código después de que la prueba haya sido superada, y vuelva a ejecutar la prueba para asegurarse de que sigue pasando contra el código refactorizado.
- Se repite este proceso para el siguiente pequeño fragmento de código, ejecutando las pruebas anteriores así como las pruebas añadidas.

Las pruebas escritas son principalmente de nivel unitario y se centran en el código, aunque también pueden escribirse pruebas a nivel de integración o de sistema. El desarrollo guiado por pruebas ganó su popularidad gracias a la Programación Extrema [Beck02], pero también se utiliza en otras metodologías ágiles y, en ocasiones, en ciclos de vida secuenciales. Ayuda a los desarrolladores a concentrarse en resultados esperados claramente definidos. Las pruebas se automatizan y se utilizan en la integración continua.

Desarrollo Guiado por Pruebas de Aceptación

El desarrollo guiado por pruebas de aceptación [Adzic09] define los criterios de aceptación y las pruebas durante la creación de las historias de usuario (véase el apartado 1.2.2). 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. El proceso de desarrollo guiado por prueba de aceptación se explica en la sección 3.3.2.

El desarrollo guiado por pruebas de aceptación crea pruebas reutilizables para las pruebas de regresión. Hay herramientas específicas que apoyan la creación y ejecución de dichas pruebas, a menudo dentro del proceso de integración continua. Estas herramientas pueden conectarse a las capas de datos y servicios de la aplicación, lo que permite ejecutar las pruebas a nivel de sistema o de aceptación. El desarrollo guiado por prueba de aceptación permite resolver rápidamente los defectos y validar el comportamiento de las prestaciones. Ayuda a determinar si se cumplen los criterios de aceptación de la prestación.

Desarrollo Guiado por el Comportamiento

El desarrollo guiado por el comportamiento [Chelimsky10] 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.

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

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

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

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 (véase [ISTQB_FL_SYL], Sección 2.2). 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). Por lo general, las pruebas unitarias y de nivel de integración se automatizan y se crean utilizando herramientas basadas en API. En los niveles de sistema y aceptación, las pruebas automatizadas se crean utilizando herramientas basadas en la IGU. El concepto de pirámide de prueba se basa en el principio de control de calidad y pruebas tempranas (es decir, eliminar los defectos lo antes posible en el ciclo de vida).

3.1.3 Cuadrantes de Prueba, Niveles de Prueba y Tipos de Prueba

Los cuadrantes de prueba, definidos por Brian Marick [Crispin08], 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. Este modelo también proporciona una forma de diferenciar y describir los tipos de pruebas a todos los implicados, incluidos los desarrolladores, probadores y representantes de negocio.

En los cuadrantes de prueba, éstas pueden estar orientadas al negocio (usuario) o a la tecnología (desarrollador). Algunas pruebas apoyan el trabajo realizado por el equipo ágil y confirman el comportamiento del software. Otras pruebas pueden verificar el producto. Las pruebas pueden ser totalmente manuales, 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, ejemplos, pruebas de historia, prototipos de experiencia de usuario y simulaciones. Estas pruebas comprueban los criterios de aceptación y pueden ser manuales o automatizadas. Suelen crearse durante el desarrollo de la historia de usuario y así mejoran la calidad de las historias. Son útiles para crear conjuntos de pruebas de regresión automatizadas.
- El cuadrante Q3 es el nivel de aceptación de sistema o usuario, de cara al negocio, y contiene pruebas que critican el producto, utilizando escenarios y datos realistas. Este cuadrante contiene pruebas exploratorias, escenarios, flujos de procesos, pruebas de usabilidad, pruebas de aceptación de usuario, pruebas alfa y pruebas beta. 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, carga, estrés y escalabilidad, pruebas de seguridad, mantenibilidad, gestión de la memoria, compatibilidad e interoperabilidad, migración de datos, infraestructura y pruebas de recuperación. Estas pruebas suelen estar automatizadas.

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

3.1.4 El Rol de un Probador

A lo largo de este programa de estudio, se ha hecho una referencia general a los métodos y técnicas ágiles, y al rol de un probador dentro de varios ciclos de vida ágiles. Esta subsección examina específicamente el rol de un probador en un proyecto que sigue un ciclo de vida Scrum [Aalst13].

Trabajo en Equipo

El trabajo en equipo es un principio fundamental en el desarrollo ágil. Ágil enfatiza el enfoque de equipo completo formado por desarrolladores, probadores y representantes de negocio que trabajan juntos. 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. El equipo trabaja conjuntamente en la estrategia de prueba, la planificación de prueba, la especificación de prueba, la ejecución de prueba, la evaluación de prueba y el informe de resultados de prueba.
- Autoorganizado: El equipo puede estar formado sólo por desarrolladores, pero, como se indica en la sección 2.1.5, 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 otros equipos, 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 (desarrolladores, probadores y Scrum Master), 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 (véase la sección 2.2.1).
- 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. Esto suele hacerse proporcionando información a los implicados sobre el proceso de prueba.
- Abierto a la retroalimentación: La retroalimentación es un aspecto importante para tener éxito en cualquier proyecto, especialmente en los proyectos ágiles. Las retrospectivas permiten a los equipos aprender de los éxitos y de los fallos.
- Resiliente: La prueba debe ser capaz de responder al cambio, como todas las demás actividades en los proyectos ágiles.

Estas mejores prácticas maximizan la probabilidad de éxito de la prueba en los proyectos Scrum.

Esprint Cero

El esprint cero es la primera iteración del proyecto en la que tienen lugar muchas actividades de preparación (véase el apartado 1.2.5). El probador colabora con el equipo en las siguientes actividades durante esta iteración:

- Identificar el alcance del proyecto (es decir, la lista de trabajo acumulado del producto).
- Crear una arquitectura inicial del sistema y prototipos de alto nivel.

- Planificar, adquirir e instalar las herramientas necesarias (por ejemplo, para la gestión de pruebas, la gestión de defectos, la automatización de pruebas y la integración continua).
- Crear una estrategia de prueba inicial para todos los niveles de la prueba, abordando (entre otros temas) el alcance de la prueba, los riesgos técnicos, los tipos de prueba (véase la sección 3.1.3) y los objetivos de cobertura.
- Realice un análisis del riesgo de calidad inicial (véase el apartado 3.2.1).
- Definir las métricas de prueba para medir el proceso de prueba, el avance de la prueba en el proyecto y la calidad del producto.
- Especificar la definición de hecho.
- Cree el tablero de tareas (véase el apartado 2.2.1).
- Definir cuándo continuar o detener la prueba antes de entregar el sistema al cliente.

El esprint cero establece la dirección de lo que las pruebas deben lograr y cómo deben lograrlo a lo largo de los esprints.

Integración

En los proyectos ágiles, el objetivo es entregar valor al cliente de forma continua (preferiblemente en cada esprint). 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 de la prueba para la entrega y cada esprint debe abordar los temas tratados en la sección 1.2.5.

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. Además, se debe hacer un seguimiento de cualquier problema de pruebas para mantener un flujo de prueba constante.

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 (por ejemplo, un probador y un desarrollador, dos probadores o un probador y un propietario del producto) 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, empezando por pruebas sencillas y avanzando hacia otras más complejas.

- Mapa Mental: Los mapas mentales son una herramienta útil a la hora de probar [Crispin08]. Por ejemplo, los probadores pueden utilizar los mapas mentales para identificar las sesiones de prueba que deben realizar, para mostrar las estrategias de prueba y para describir los datos de prueba.

Estas prácticas se suman a otras que se tratan en este programa de estudio y en el capítulo 4 del programa de estudio de nivel básico [ISTQB_FL_SYL].

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. Los probadores de los proyectos ágiles pueden utilizar los mismos tipos de técnicas que se emplean en los proyectos tradicionales para identificar los riesgos de calidad (o los riesgos de producto), evaluar el nivel de riesgo asociado, estimar el esfuerzo necesario para reducir suficientemente esos riesgos y, a continuación, mitigarlos mediante el diseño, la implementación y la ejecución de las pruebas. Sin embargo, dadas las cortas iteraciones y el ritmo de cambio de los proyectos ágiles, se requieren algunas adaptaciones de esas técnicas.

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. Esto incluye la determinación de la cantidad adecuada de esfuerzo que se debe asignar para cubrir cada condición con pruebas, y la secuenciación de las pruebas resultantes de forma que se optimice la efectividad y la eficiencia del trabajo de prueba que se va a realizar. 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.

El riesgo es la posibilidad de que se produzca un resultado o evento negativo o indeseable. El nivel de riesgo se encuentra evaluando la probabilidad de que ocurra el riesgo y el impacto del mismo. Cuando el efecto principal del problema potencial es la calidad del producto, los problemas potenciales se denominan riesgos de calidad o riesgos de producto. Cuando el efecto principal del problema potencial es sobre el éxito del proyecto, los problemas potenciales se denominan riesgos de proyecto o riesgos de planificación [Black07] [vanVeenendaal12].

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.

Algunos ejemplos de riesgos de calidad para un sistema son:

- Cálculos incorrectos en los informes (un riesgo funcional relacionado con la exactitud).
- Respuesta lenta a la entrada del usuario (un riesgo no funcional relacionado con la eficiencia y el tiempo de respuesta).

- Dificultad para entender las pantallas y los campos (un riesgo no funcional relacionado con la usabilidad y la capacidad de ser entendido).

Como se ha mencionado anteriormente, una iteración comienza con la planificación de la iteración, que culmina con las tareas estimadas en un tablero de tareas. 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. Las tareas asociadas a los riesgos más bajos deberían empezar más tarde e implicar un menor 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.

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

A lo largo del proyecto, el equipo debe permanecer atento a la información adicional que pueda cambiar el conjunto de riesgos y/o el nivel de riesgo asociado a los riesgos de calidad conocidos. Debe producirse un ajuste periódico del análisis del riesgo de calidad, que dé lugar a ajustes en las pruebas. Los ajustes incluyen la identificación de nuevos riesgos, la reevaluación del nivel de los riesgos existentes y la evaluación de la efectividad de las actividades de mitigación del riesgo.

Los riesgos de calidad también pueden mitigarse antes de que comience la ejecución de pruebas. Por ejemplo, si se encuentran problemas con las historias de usuario durante la identificación del riesgo, el equipo del proyecto puede revisar a fondo las historias de usuario como estrategia de mitigación.

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. El propietario de producto o el cliente lee una historia de usuario a los estimadores. Cada estimador dispone de una baraja de cartas con valores similares a la secuencia de Fibonacci (es decir, 0, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...) o cualquier otra progresión de su elección (por ejemplo, tallas de camisas que van de la extra-pequeña a la extra-extra-grande). Los valores representan el número de puntos de historia, días de esfuerzo u otras unidades en las que el equipo estima. Se recomienda la secuencia de Fibonacci porque los números de la secuencia reflejan que la incertidumbre crece proporcionalmente con el tamaño de la historia. 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. Aspectos como el esfuerzo de desarrollo y pruebas, la complejidad de la historia y el alcance de la prueba juegan un papel en la estimación. 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, antes de iniciar la sesión de póker de planificación. Cuando se ha discutido completamente la prestación, cada estimador selecciona en privado una carta para representar su estimación. A continuación, se revelan todas las cartas al mismo tiempo. Si todos los estimadores han seleccionado el mismo valor, éste se convierte en la estimación. Si no es así, los estimadores discuten las diferencias en las estimaciones, tras lo cual se repite la ronda de póquer hasta que se llegue a un acuerdo, ya sea por consenso o aplicando reglas (por ejemplo, utilizar la mediana, utilizar la puntuación más alta) para limitar el número de rondas de póquer. Estas discusiones garantizan una estimación fiable del esfuerzo necesario para completar los elementos de la lista de trabajo acumulado del producto solicitados por el propietario del mismo y ayudan a mejorar el conocimiento colectivo de lo que hay que hacer [Cohn04].

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 (véase el apartado 1.2.2). Los requisitos no funcionales, como la usabilidad y el rendimiento, también son importantes y pueden especificarse como historias de usuario únicas o conectadas a otras historias de usuario funcionales. Los requisitos no funcionales pueden seguir un formato o estándar predefinido, como [ISO25000], o un estándar específico del sector.

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 (contexto, configuraciones del sistema y comportamiento del usuario).
- Información sobre defectos de proyectos existentes y anteriores
- Una categorización de los defectos en una taxonomía de defectos.
- Estándares aplicables (por ejemplo, [DO-178B] para el software del sector aeronáutico).
- Riesgos de calidad (véase el apartado 3.2.1).

Durante cada iteración, los desarrolladores crean un código que implementa las funciones y prestaciones descritas en las historias de usuario, con las características de calidad pertinentes, y este código se verifica y valida mediante pruebas de aceptación. Para que se pueda probar, los criterios de aceptación deben abordar los siguientes temas cuando sean relevantes [Wiegers13]:

- 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. Las características de calidad más comunes son el rendimiento, la fiabilidad, la usabilidad, etc.
- 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 (por ejemplo, los procedimientos utilizados por una compañía de seguros para gestionar las reclamaciones de seguros).
- 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 (interfaz de usuario, interfaz con otros sistemas, etc.).
- Restricciones: Cualquier restricción de diseño e implementación que limite las opciones del desarrollador. Los dispositivos con software integrado deben respetar a menudo restricciones físicas como el tamaño, el peso y las conexiones de la interfaz.
- 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 (por ejemplo, el código postal en una dirección de correo de EE.UU.).

Además de las historias de usuario y sus criterios de aceptación asociados, hay otra información relevante para el probador, que incluye:

- Cómo se supone que funciona y se utiliza el sistema.
- Las interfaces del sistema que se pueden utilizar/acceder para probar el sistema.
- Si el soporte actual de las herramientas es suficiente.
- Si el probador tiene suficientes conocimientos y competencia para realizar las pruebas necesarias.

Los probadores descubrirán a menudo la necesidad de información adicional (por ejemplo, cobertura de código) a lo largo de las iteraciones y deberán trabajar en colaboración con el resto de los miembros del equipo ágil para obtener esa información. La información relevante juega un papel importante a la hora de determinar si una actividad concreta puede considerarse hecha. Este concepto de definición de hecho es fundamental en los proyectos ágiles y se aplica de varias formas diferentes, como se analiza en las siguientes subsecciones.

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 (clasificado en función de la prioridad y la severidad).
  - Ninguna deuda técnica inaceptable conocida en el diseño y el código [Jones11].
  - 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 (por ejemplo, el rendimiento).
- 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 (por ejemplo, rendimiento, robustez, fiabilidad).
  - Pruebas realizadas en un entorno o entornos similares a los de producción, incluyendo todo el hardware y el software para todas las configuraciones compatibles, en la medida de lo posible.
  - 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 (priorizados según el riesgo e importancia).

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. La adecuación de la cobertura se determina en función de lo que es nuevo o ha cambiado, de su complejidad y tamaño, y de los riesgos de fallo asociados.

- Calidad: La intensidad de los defectos (por ejemplo, cuántos defectos se encuentran por día o por transacción), la densidad de defectos (por ejemplo, el número de defectos encontrados en comparación con el número de historias de usuario, el esfuerzo y/o los atributos de calidad), el número estimado de defectos restantes están dentro de los límites aceptables, las consecuencias de los defectos no resueltos y restantes (por ejemplo, la severidad y la prioridad) 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 (es decir, el coste de desarrollo y mantenimiento calculado debe ser considerablemente inferior a las ventas totales previstas del producto). La parte principal del coste del ciclo de vida suele proceder del mantenimiento después de la entrega del producto, debido al número de defectos que consiguen acceder a producción.

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 [Adzic09] y pueden ser manuales o automatizados. El primer paso es un taller de especificación en el que los desarrolladores, probadores y representantes de negocio analizan, discuten y redactan la historia de usuario. Durante este proceso se corrige cualquier carencia, ambigüedad o error en la historia de usuario.

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. Estos ejemplos ayudarán al equipo a implementar correctamente la historia de usuario. Dado que los ejemplos y las pruebas son lo mismo, estos términos suelen utilizarse indistintamente. El trabajo comienza con ejemplos básicos y preguntas abiertas.

Normalmente, las primeras pruebas son las positivas, que confirman el comportamiento correcto sin condiciones de excepción o error, y que comprenden la secuencia de actividades que se ejecutan si todo va como se espera. Una vez realizadas las pruebas del camino positivo, el equipo debe escribir las pruebas del camino negativo y cubrir también los atributos no funcionales (por ejemplo, el rendimiento, la usabilidad). Las pruebas se expresan de forma que cualquier implicado pueda entenderlas, con frases en lenguaje natural que incluyan las precondiciones necesarias, si las hay, las entradas y las salidas relacionadas.

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

3.3.3 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. (Algunas pruebas, como las exploratorias y otras basadas en la experiencia, se crean más tarde, durante la ejecución de pruebas, como se explica en el apartado 3.3.4.) Los probadores pueden aplicar técnicas de diseño de prueba de caja negra tradicionales, como el análisis de particiones de equivalencia, el análisis del valor frontera, las tablas de decisión y las pruebas de transición de estado para crear estas pruebas. Por ejemplo, el análisis del valor frontera podría utilizarse para seleccionar los valores de prueba cuando un cliente tiene un número limitado de elementos que puede seleccionar para comprar.

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

Para más información sobre el uso de técnicas de diseño de prueba de caja negra, consulte el programa de estudio de nivel básico [ISTQB_FL_SYL] y el programa de estudio de analista de pruebas de nivel avanzado [ISTQB_ALTA_SYL].

3.3.4 Prueba Exploratoria y Prueba Ágil

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. Las estrategias de prueba y la combinación de estrategias de prueba se discuten en el programa de estudio de nivel básico [ISTQB_FL_SYL].

En la prueba exploratoria, el diseño de prueba y la ejecución de prueba ocurren al mismo tiempo, guiados por un contrato de prueba establecido. Un contrato de prueba proporciona las condiciones de prueba que se deben cubrir durante una sesión de prueba con límite de tiempo. Durante la prueba exploratoria, los resultados de las pruebas más recientes sirven de guía para la siguiente prueba. Para diseñar las pruebas se pueden utilizar las mismas técnicas de caja blanca y caja negra que cuando se realizan pruebas prediseñadas.

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 (por ejemplo, "Entrar en el sistema como superusuario") y lo que sería interesante probar (pruebas positivas y negativas).
- Notas del oráculo: cómo evaluar el producto para determinar los resultados correctos (por ejemplo, captar lo que ocurre en la pantalla y compararlo con lo que está escrito en el manual del usuario).
- Variaciones: acciones y evaluaciones alternativas para complementar las ideas descritas en las actividades.

Para gestionar las pruebas exploratorias, se puede utilizar un método denominado gestión de prueba basada en sesiones. Una sesión se define como un periodo ininterrumpido de pruebas que puede durar entre 60 y 120 minutos. Las sesiones de prueba incluyen lo siguiente:

- Sesión de estudio (para saber cómo funciona).
- Sesión de análisis (evaluación de la funcionalidad o las características).
- Cobertura profunda (casos extremos), escenarios e interacciones.

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

- ¿Qué es lo más importante que hay que averiguar sobre el sistema?
- ¿De qué manera puede fallar el sistema?
- ¿Qué ocurre si.....?
- ¿Qué debería ocurrir cuando.....?
- ¿Se cumplen las necesidades, requisitos y expectativas del cliente?
- ¿Es posible instalar el sistema (y eliminarlo si es necesario) en todos los caminos de actualización admitidos?

Durante la ejecución de prueba, el probador utiliza la creatividad, la intuición, el conocimiento y la competencia para encontrar posibles problemas con el producto. El probador también debe tener un buen conocimiento y comprensión del software sometido a prueba, del dominio del negocio, de cómo se utiliza el software y de cómo establecer cuándo falla el sistema.

A la hora de probar se puede aplicar un conjunto de heurísticas. Una heurística puede orientar al probador sobre cómo realizar las pruebas y evaluar los resultados [Hendrickson]. Algunos ejemplos son:

- Fronteras
- CRUD (Create: Crear; Read: Leer; Update: Actualiza; Delete: Borrar).
- Interrupciones (por ejemplo, cierre de sesión, apagado o reinicio).

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 siguiente lista ofrece ejemplos de información que puede ser útil documentar:

- Cobertura de la prueba: qué datos de entrada se han utilizado, cuánto se ha cubierto y cuánto queda por probar.

- Notas de evaluación: observaciones durante las pruebas, si el sistema y la prestación que se prueban parecen estables, si se ha encontrado algún defecto, qué se ha planificado como siguiente paso según las observaciones actuales, y cualquier otra lista de ideas.
- Lista de riesgos/estrategia: qué riesgos se han cubierto y cuáles siguen siendo los más importantes, si se va a seguir la estrategia inicial o si es necesario modificarla.
- Cuestiones, preguntas y anomalías: cualquier comportamiento inesperado, cualquier duda sobre la eficiencia del enfoque, cualquier preocupación sobre las ideas/intentos de prueba, el entorno de prueba, los datos de prueba, la incomprensión de la función, el guion de prueba o el sistema sujeto a prueba.
- Comportamiento real: registro del comportamiento real del sistema que debe guardarse (por ejemplo, vídeo, capturas de pantalla, archivos de datos de salida).

La información registrada debe capturarse y/o resumirse en algún tipo de herramienta de gestión del estado (por ejemplo, herramientas de gestión de pruebas, herramientas de gestión de tareas, el tablero de tareas), de forma que facilite a los implicados la comprensión del estado actual de todas las pruebas realizadas.

3.4 Herramientas en Proyectos Ágiles

Las herramientas descritas en el programa de estudio de nivel básico [ISTQB_FL_SYL] son relevantes y utilizadas por los probadores en equipos ágiles. No todas las herramientas se utilizan de la misma manera y algunas herramientas tienen más relevancia para los proyectos ágiles que para los proyectos tradicionales. Por ejemplo, aunque las herramientas de gestión de la prueba, las herramientas de gestión de requisitos y las herramientas de gestión de incidencias (herramientas de seguimiento de defectos) pueden ser utilizadas por los equipos ágiles, algunos equipos ágiles optan por una herramienta integral (por ejemplo, la gestión del ciclo de vida de la aplicación o la gestión de tareas) que proporciona prestaciones relevantes para el desarrollo ágil, como tableros de tareas, gráficos de quemado e historias de usuario. Las herramientas de gestión de la configuración son importantes para los probadores de los equipos ágiles debido al elevado número de pruebas automatizadas en todos los niveles y a la necesidad de almacenar y gestionar los artefactos de prueba automatizados asociados.

Además de las herramientas descritas en el programa de estudio de nivel básico [ISTQB_FL_SYL], los probadores en proyectos ágiles también pueden utilizar las herramientas descritas en las siguientes subsecciones. Estas herramientas son utilizadas por todo el equipo para asegurar la colaboración del equipo y el intercambio de información, que son clave para las prácticas ágiles.

3.4.1 Herramientas de Gestión y Seguimiento de Tareas

En algunos casos, los equipos ágiles utilizan tableros físicos de historias/tareas (por ejemplo, pizarra blanca, pizarra de corcho) para gestionar y hacer un seguimiento de las historias de usuario, las pruebas y otras tareas a lo largo de cada esprint. Otros equipos utilizan software de gestión del ciclo de vida de la aplicación y de gestión de tareas, incluidos tableros de tareas electrónicos. 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 (a través de métricas, gráficos y paneles de control) del estado actual de cada historia de usuario, la iteración y la entrega, lo que permite a todos los implicados, incluidos los miembros de equipos distribuidos geográficamente, comprobar rápidamente el estado.
- 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. Los usos, beneficios y riesgos de estas herramientas fueron descritos anteriormente en la Sección 1.2.4.

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. Esto proporciona trazabilidad entre qué versiones del software se probaron con qué versiones particulares de las pruebas, y permite un cambio rápido sin perder la información histórica. Los principales tipos de sistemas de control de versiones son los sistemas de control de versiones centralizados y los sistemas de control de versiones distribuidos. 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. Estas herramientas también pueden ayudar a redefinir la estructura de la base de datos a medida que el producto sufre cambios durante un proyecto ágil y a refactorizar los scripts para generar los datos. Esto permite actualizar rápidamente los datos de prueba a medida que se producen cambios. Algunas herramientas de preparación de datos de prueba utilizan fuentes de datos de producción como materia prima y

utilizan scripts para eliminar o anonimizar los datos sensibles. Otras herramientas de preparación de datos de prueba pueden ayudar a validar grandes entradas o salidas de datos.

- Herramientas de carga de datos de prueba: Una vez generados los datos para las pruebas, 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. De hecho, muchas de las herramientas de generación de datos incluyen un componente de carga de datos integrado. En otros casos, también es posible la carga masiva mediante los sistemas de gestión de bases de datos.
- Herramientas de ejecución de pruebas automatizadas: Existen herramientas de ejecución de prueba que están más alineadas con las pruebas ágiles. Existen herramientas específicas, tanto comerciales como de código abierto, para dar soporte a los primeros enfoques de pruebas, como el desarrollo guiado por el comportamiento, el desarrollo guiado por pruebas y el desarrollo guiado por pruebas de aceptación. Estas herramientas permiten a los probadores y al personal de negocio expresar el comportamiento esperado del sistema en tablas o en lenguaje natural 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. Esto es útil cuando se encuentra un defecto, ya que se han capturado las acciones realizadas antes de que se produjera el fallo y pueden utilizarse para informar del defecto a los desarrolladores. El registro de los pasos realizados en una sesión de prueba exploratoria puede resultar beneficioso si la prueba se incluye finalmente en el conjunto de pruebas de regresión automatizadas.

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. El aprovisionamiento de un nuevo servidor o la restauración de un servidor es más eficiente con las capacidades de instantáneas integradas en la mayoría de las herramientas de virtualización. Algunas herramientas de gestión de la prueba utilizan ahora tecnologías de virtualización para tomar instantáneas de los servidores en el momento en que se detecta una falta, lo que permite a los probadores compartir la instantánea con los desarrolladores que investigan la falta.

7 Referencias

4.1 Estándares

- [DO-178B] RTCA/FAA DO-178B, Software Considerations in Airborne Systems and Equipment Certification, 1992.
- [ISO25000] ISO/IEC 25000:2005, Software Engineering - Software Product Quality Requirements and Evaluation (SQuaRE), 2005.

4.2 Documentos del ISTQB

- [ISTQB_ALTA_SYL] ISTQB Advanced Level Test Analyst Syllabus, Version 2012.
- [ISTQB_ALTM_SYL] ISTQB Advanced Level Test Manager Syllabus, Version 2012.
- [ISTQB_FA_OVIEW] ISTQB Foundation Level Agile Tester Overview, Version 1.0.
- [ISTQB_FL_SYL] ISTQB Foundation Level Syllabus, Version 2011.

4.3 Libros

- [Aalst13] Leo van der Aalst and Cecile Davis, “TMap NEXT® in Scrum,” ICT-Books.com, 2013. [Adzic09] Gojko Adzic, “Bridging the Communication Gap: Specification by Example and Agile Acceptance Testing,” Neuri Limited, 2009.
- [Anderson13] David Anderson, “Kanban: Successful Evolutionary Change for Your Technology Business,” Blue Hole Press, 2010.
- [Beck02] Kent Beck, “Test-driven Development: By Example,” Addison-Wesley Professional, 2002. [Beck04] Kent Beck and Cynthia Andres, “Extreme Programming Explained: Embrace Change, 2e” Addison-Wesley Professional, 2004.
- [Black07] Rex Black, “Pragmatic Software Testing,” John Wiley and Sons, 2007.
- [Black09] Rex Black, “Managing the Testing Process: Practical Tools and Techniques for Managing Hardware and Software Testing, 3e,” Wiley, 2009.
- [Chelimsky10] David Chelimsky et al, “The RSpec Book: Behavior Driven Development with Rspec, Cucumber, and Friends,” Pragmatic Bookshelf, 2010.
- [Cohn04] Mike Cohn, “User Stories Applied: For Agile Software Development,” Addison-Wesley Professional, 2004.
- [Crispin08] Lisa Crispin and Janet Gregory, “Agile Testing: A Practical Guide for Testers and Agile Teams,” Addison-Wesley Professional, 2008.
- [Goucher09] Adam Goucher and Tim Reilly, editors, “Beautiful Testing: Leading Professionals Reveal How They Improve Software,” O'Reilly Media, 2009.
- [Jeffries00] Ron Jeffries, Ann Anderson, and Chet Hendrickson, “Extreme Programming Installed,” Addison-Wesley Professional, 2000.
- [Jones11] Capers Jones and Olivier Bonsignour, “The Economics of Software Quality,” Addison-Wesley Professional, 2011.
- [Linz14] Tilo Linz, “Testing in Scrum: A Guide for Software Quality Assurance in the Agile World,” Rocky Nook, 2014.
- [Schwaber01] Ken Schwaber and Mike Beedle, “Agile Software Development with Scrum,” Prentice Hall, 2001.
- [vanVeenendaal12] Erik van Veenendaal, “The PRISMA approach”, Uitgeverij Tutein Nolthenius, 2012.
- [Wiegers13] Karl Wiegers and Joy Beatty, “Software Requirements, 3e,” Microsoft Press, 2013.

4.4 Términos Ágiles

Las palabras clave que se encuentran en el Glosario del ISTQB se identifican al principio de cada capítulo. Para los términos comunes de Agile, nos hemos basado en los siguientes recursos de Internet bien aceptados que proporcionan definiciones.

- http://guide.Agilealliance.org/
- http://whatis.techtargetcom/glossary
- http://www.scrumalliance.org/

Animamos a los lectores a consultar estos sitios si encuentran en este documento términos relacionados con Agile que no les resulten familiares. Estos enlaces estaban activos en el momento de la entrega de este documento.

4.5 Otras Referencias

Las siguientes referencias remiten a información disponible en Internet y en otros lugares. Aunque estas referencias fueron comprobadas en el momento de la publicación de este programa de estudio, el ISTQB no se hace responsable si las referencias ya no están disponibles.

- [Agile Alliance Guide] Various contributors, http://guide.Agilealliance.org/.
- [Agilemanifesto] Various contributors, www.agilemanifesto.org.
- [Hendrickson]: Elisabeth Hendrickson, “Acceptance Test-driven Development,” testobsessed.com/2008/12/acceptance-test-driven-development-atdd-an-overview.
- [INVEST] Bill Wake, “INVEST in Good Stories, and SMART Tasks,” xp123.com/articles/invest- in-good-stories-and-smart-tasks.
- [Kubaczkowski] Greg Kubaczkowski and Rex Black, “Mission Made Possible,” www.rbcs-us.com/images/documents/Mission-Made-Possible.pdf.

8 Índice

G

H

I

K

L

M

O

P