Certified Practitioner in Agile Testing (CPAT)

Plan de estudios

Version 1.02 Spanish

Released March-2020

Aviso de derechos de autor

Gracias a los principales autores

- Bas Kruip, Joost Voskuil, Klaartje van Zwoll, Jeroen van Seeters, Huib Schoots.

Gracias a los coautores

- Carlo van Driel, Jayapradeep Jiothis.

Gracias al comité de revisión

Alexis Herrera, Alfonso Fernández, Ana Laura Ochoa Moreno, Arun Janglie, Aurelio Gandarillas, Ángel Rayo Acevedo, Christine Green, Daniel Castillo Garcia, Daniel Leo Lopez Romero, Dwarakanath Babu KLSD, Emilie Potin-Suau, Erik van Veenendaal, Fabiola Mero, Gustavo Márquez Sosa, Héctor Ruvalcaba, Isaac Marcelo Malamud Kobrinsky, Ismael Betancourt, Javier Chávez, Javier Jesús Gutiérrez Rodríguez, Jordi Fernandez, José Antonio Rodriguez, Julian Baars, Julie Gardiner,Julio Córdoba Retana, Jochem Gross, Kaan Sanli, Kyle Alexander Siemens, Laksh Ranganathan, Luisa Morales Gómez-Tejedor, Márcia Araújo Coelho, Marco Fidel Peña Valbuena, Marelis V. Pérez García, Mario Alvarez Gómez, Melissa Pontes, Miaomiao Tang, Miguel Angel De León Trejo, Miranda Kerpel, Nadia Soledad Cavalleri, Patricia Osorio Aristizabal, Paul Mowat, Richard Seidl, Rik Marselis, Rogier Ammerlaan, Ruth Margaret Florian Caipa, Sammy Kolluru, Samuel Ouko, Sebastiaan Vreedenburgh, Sergio von Borries, Shashikumar Singh, Silvia Nane, Søren Wassard, Thomas Cagley, Tim Moore, Ulises Gonzalez, Valeria Cocco, Vladimir Berrio, Wim Decoutere & Yaara Egger.

Historial de revisión

Versión	Fecha	Comentarios
0.16	Diciembre 2019	Lanzamiento Inicial Beta inglés
1.00	Enero 2020	Lanzamiento Inicial después de revisión inglés
1.01	Enero 2020	Integración comentarios de revisión inglés
1.01SPA Beta	Febrero 2020	Lanzamiento Inicial Beta español
1.01SPA	Marzo 2020	Ultima revisión
1.02SPA	Marzo 2020	Se añadió referencia de los modelos de Davis Hestenes en el capítulo 2.4

Tabla de contenido

Resultados de las actividades (RAs)

Los resultados de las actividades (RAs) son una breve declaración de lo que se espera que hayas aprendido después de la formación.

RA-1	Comprender los principios de Agile y Scrum en general
RA-2	Comprender los principios de Agile y Scrum en relación con las pruebas
RA-3	Comprender el papel de un tester en un entorno ágil
RA-4	Tener una visión de las habilidades de las personas que son esenciales para formar parte de un equipo multidisciplinario y para colaborar eficazmente con los desarrolladores, analistas y propietarios del producto.
RA-5	Ser capaz de discutir y determinar prácticamente los riesgos utilizando múltiples técnicas
RA-6	Comprender la relación entre los riesgos, los hechos, las preguntas, la intuición, la exploración y las pruebas
RA-7	Definir una estrategia de prueba adaptada para trabajar en un entorno ágil
RA-8	Practicar creando una estrategia de prueba para un producto real
RA-9	Comprender y utilizar las pruebas exploratorias para acelerar sus pruebas
RA-10	Aplicar las pruebas exploratorias al software real
RA-11	Comprender y usar Automatización de pruebas de forma sostenible (mantenible y reutilizable)
RA-12	Ser capaz de describir la historia de la prueba
RA-13	Ser capaz de presentar informes efectivos sobre las pruebas a la administración y a otros interesados.
RA-14	Ser capaz de mejorar las especificaciones para añadir más valor
RA-15	Usar sus habilidades de pensamiento crítico para mejorar la (calidad de) el producto y/o los procesos
RA-16	Comprender y practicar los escenarios de retroalimentación
RA-17	Comprender el aprendizaje y el papel de las retrospectivas en el aprendizaje
RA-18	Practicar el aprendizaje mediante la aplicación de retrospectivas

Objetivos de aprendizaje/niveles cognitivos de conocimiento

Los objetivos de aprendizaje (OAs) son declaraciones breves que describen lo que se espera que sepas después de estudiar cada capítulo. Los OAs se definen con base a la taxonomía de Bloom de la siguiente manera:

Definiciones	**K1 Recordar**	**K2 Comprender**	**K3 Aplicar**
Definición de Bloom	Exhibir la memoria de material previamente aprendido recordando hechos, términos, conceptos básicos y respuestas.	Demostrar la comprensión de los hechos e ideas organizando, comparando, traduciendo, interpretando, dando descripciones y declarando las ideas principales.	Resolver problemas a nuevas situaciones aplicando de manera diferente los conocimientos, hechos, técnicas y reglas adquiridos.

Verbos (ejemplos)	Recordar Elegir Definir Encontrar Comparar Relacionar Seleccionar	Resumir Generalizar Clasificar Comparar Contrastar Demostrar Interpretar Repetir	Implementar Ejecutar Utilizar Aplicar Planificar Seleccionar

Para más detalles de la taxonomía de Bloom, por favor, consulte **[BT1]** y **[BT2]** en las Referencias.

Objetivos Prácticos

Los objetivos prácticos (OPs) son declaraciones breves que describen lo que se espera que realice o ejecute para comprender el aspecto práctico del aprendizaje. Los OPs se definen de la siguiente manera:

- OP-0: Vista en vivo de un ejercicio o video grabado.
- OP-1: Ejercicio guiado. Los alumnos siguen la secuencia de pasos que realiza el formador.
- OP-2: Ejercicio con pistas. Ejercicio a ser resuelto por el aprendiz, utilizando pistas proporcionadas por el formador.
- OP-3: Ejercicios no guiados sin pistas.

Prerrequisitos

Obligatorio

- Ninguno

Recomendado

- Algún certificado de Agile o Scrum como PSM o CSM o ASF o por lo menos leer la guía de Scrum.
- Conocimientos básicos de pruebas en general (ISTQB-CTFL o ISTQB-CFTL-Agile Tester).
- Tener al menos 1 año de experiencia laboral en Agile y en Testing.
- Leer un libro sobre Pruebas Ágiles como 'Agile Testing'**[JL1]** y 'More Agile Testing'**[JL1]**.

Capítulo 1 - Introducción a las pruebas ágiles

En la introducción, definiremos la terminología que rodea a las pruebas, los riesgos, Ágil, Scrum y Pruebas Ágiles.

Palabras clave

Pruebas, riesgo del producto, riesgo del proyecto, riesgo de calidad, ágil, manifiesto, conocidos conocidos, desconocido conocidos, conocidos desconocidos, desconocidos desconocidos, pruebas ágiles, Scrum.

OA-1.1	K1	Comprender la definición de prueba (testing)
OA-1.2	K1	Recordar la estructura de los diversos riesgos
OA-1.3	K1	Recordar el "Manifiesto por el Desarrollo Ágil de Software"
OA-1.4	K2	Explicar el significado del Manifiesto y los principios
OA-1.5	K2	Replantear el Manifiesto en riesgos, basado en "Hay conocimientos conocidos" de Rumsfeld
OA-1.6	K2	Comprender y explicar la definición de prueba y especialmente de prueba ágil
OA-1.7	K2	Comprender el marco de Scrum
OP-1.1	OP-3	Compartir las definiciones actuales de Scrum entre el grupo
OA-1.8	K2	Explicar el marco de Scrum y el papel de un tester en el equipo
OA-1.9	K1	Recordar los roles del product owner, el scrum máster y el equipo de scrum
OA-1.10	K1	Recordar el uso de puntos de historia y estimación y puntos de prueba vs. incluir estimaciones de prueba
OA-1.11	K2	Explicar la perspectiva de la prueba en la planificación del poker y el papel de la prueba en la planificación del sprint
OA-1-12	K1	Recordar la planificación impulsada por la velocidad y el compromiso
OA-1.13	K2	Comprender las estimaciones de las pruebas
OA-1.14	K2	Explicar el papel de la prueba en las sesiones de refinamiento
OA-1.15	K2	Explicar el papel de la prueba en las retrospectivas
OA-1.16	K2	Comprender el papel de la prueba en un programa de aumento / planificación de la liberación
OA-1.17	K1	Aprender algunos anti-patrones Scrum comúnmente usados como la iteración de prueba N+1 o las “hardening iterations” (iteraciones de endurecimiento) para resolver la deuda técnica
OA-1.18	K2	Comprender el cambio del papel del tester en un contexto ágil
OA-1.19	K2	Ser capaz de explicar cuál es el papel de los testers en un equipo ágil
OA-1.20	K2	Entender el papel de un tester como el conductor de calidad clave en un equipo ágil
OA-1.21	K2	Comprender las habilidades que son esenciales para un tester en un contexto ágil
OP-1.2	OP-3	Obtener las definiciones de los estudiantes para iniciar una discusión
OA-1.23	K2	Demostrar la definición de pensamiento crítico
OA-1.24	K2	Demostrar la definición de una reclamación o afirmación
OP-1.3	OP-1	Practicar con las reclamaciones, conclusiones y construcciones para comprender la complejidad
OA-1.25	K2	Comprender y reconocer las falacias
OA-1.26	K2	Comprender los errores de pensamiento y cómo influyen en el juicio de las personas
OA-1.27	K3	Aplicar todas las definiciones que juegan un papel en el pensamiento crítico

OP-1.4	OP-2	Practicar el pensamiento crítico usando argumentos, reclamaciones, construcciones, falacias y errores de pensamiento
OA-1.28	K2	Comprender la definición de una retrospectiva
OA-1.29	K2	Explicar el propósito de las retrospectivas
OA-1.30	K2	Demostrar la necesidad de las retrospectivas y el papel de un tester
OA-1.31	K3	Ser capaz de implementar retrospectivas en un equipo
OA-1.32	K1	Poder seleccionar diferentes tipos de retrospectivas
OP-1.5	OP-3	Definir cómo un juego serio puede ser usado para una retrospectiva

1.1 Definición de prueba

OA-1.1	K1	Comprender la definición de prueba
OA-1.2	K1	Recordar la estructura de los diversos riesgos

Las pruebas dan una **visión** de los **riesgos**, donde el riesgo es cualquier amenaza al **valor del producto** que se va a entregar a el cliente.

Los riesgos pueden ser riesgos de producto que amenazan el valor del producto o riesgos de proyecto que amenazan la velocidad de entrega del producto o, en definitiva, el propio producto.

1.2 ¿Qué es ágil?

OA-1.3	K1	Recordar el "Manifiesto por el Desarrollo Ágil de Software"
OA-1.4	K2	Explicar el significado del Manifiesto y los principios

Ágil es la capacidad de crear y responder al cambio. Es una forma de lidiar y en última instancia, tener éxito en un entorno incierto y turbulento **[AG1]**.

El manifiesto por el desarrollo Ágil de Software **[AM1]** consta de 4 valores y 12 principios. Así se explican y discuten los 4 valores:

"Estamos descubriendo mejores formas de desarrollar software haciéndolo y ayudando a otros a hacerlo. A través de este esfuerzo hemos llegado a valorar"

- Individuos e interacciones **sobre** procesos y herramientas
- Software trabajando **sobre** documentación completa
- Colaboración con el cliente **sobre** la negociación del contrato
- Responder a los cambios **sobre** el seguimiento de un plan

"Es decir, mientras que hay valor en los elementos de la derecha, nosotros valoramos más los elementos de la izquierda"

Explicados y discutidos están los 12 principios, ya que son igualmente importantes para que la gente tenga la mentalidad correcta:

"Seguimos estos principios"

1. Nuestra mayor prioridad es satisfacer al cliente a través de la entrega temprana y continua de un software valioso.

2. Los cambios en los requisitos son bienvenidos, incluso en las últimas fases de desarrollo. Los procesos ágiles aprovechan el cambio para la ventaja competitiva del cliente.
3. Entregar con frecuencia programas informáticos que funcionen, desde un par de semanas hasta un par de meses, con preferencia al plazo más corto.
4. Los responsables del negocio y los desarrolladores deben trabajar juntos diariamente durante todo el proyecto.
5. Construir proyectos alrededor de individuos motivados, dándoles el entorno y el apoyo a sus necesidades y la confianza para hacer el trabajo.
6. El método más eficiente y efectivo de transmitir información a un equipo de desarrollo es la conversación cara a cara.
7. Un software que funcione es la principal medida del progreso.
8. Los procesos ágiles promueven el desarrollo sostenible. Los patrocinadores, los desarrolladores y los usuarios deben poder mantener un ritmo constante indefinidamente.
9. La atención continua a la excelencia técnica y al buen diseño aumenta la agilidad.
10. La simplicidad -el arte de maximizar la cantidad de trabajo no hecho- es esencial.
11. Las mejores arquitecturas, requisitos y diseños surgen de equipos autoorganizados.
12. A intervalos regulares, el equipo reflexiona sobre cómo ser más efectivo, luego sintoniza y ajusta su comportamiento en consecuencia.

1.3 Vincular ágil con las pruebas y los riesgos

OA-1.5	K2	Replantear el Manifiesto Ágil en riesgos, basado en "Hay conocimientos conocidos" de Rumsfeld

Rumsfeld declaró: "Los informes que dicen que algo no ha sucedido siempre me interesan, porque como sabemos, hay conocimientos conocidos (Known knowns); hay cosas que sabemos que sabemos. También sabemos que hay conocidos desconocidos (known unknowns); es decir, sabemos que hay algunas cosas que no sabemos. Pero también hay desconocidos desconocidos (unknown unknowns)— los que no sabemos que no sabemos. Y si uno mira a lo largo de la historia de nuestro país y de otros países libres, es esta última categoría la que tiende a ser la más difícil **[RF1]**.

La idea de los desconocidos desconocidos fue creada en 1955 por dos psicólogos estadounidenses, Joseph Luft (1916-2014) y Harrington Ingham (1916-1995) en su desarrollo de la ventana de Johari. La utilizaron como una técnica para ayudar a las personas a comprender mejor su relación con ellos mismos y con los demás.

Estas ideas están vinculadas a importantes pilares de la Prueba ágil:

- Hechos
- Preguntas
- Intuición/El inconsciente
- Exploración

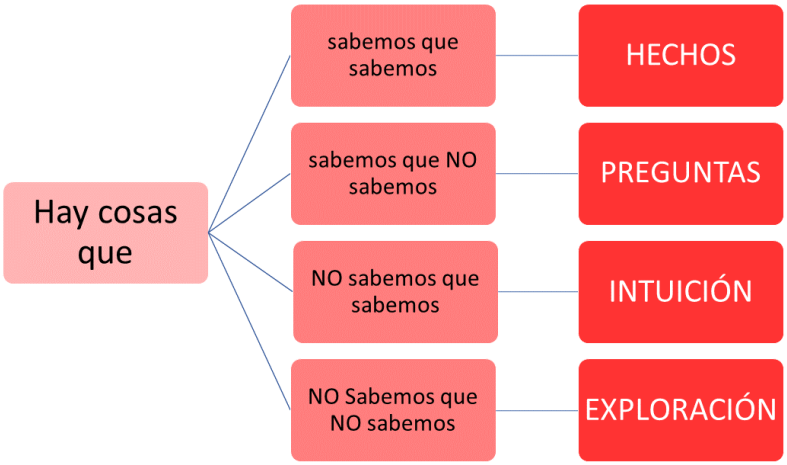


Gráfico 1

Lo que es válido para ágil en general es, por supuesto, también válido para las pruebas. Aquí también deben cubrirse todas las áreas.
Los riesgos son (propiedad de) "situaciones" que pueden presentar un peligro / amenaza el valor de los productos a entregar. El trabajo de los testers es identificar esos peligros y amenazas. El campo total de esas "amenazas" consiste en 4 subcategorías.
Todas ellas en un sentido puro requieren un enfoque diferente: en un sentido más amplio, aquellas cosas que conocemos requieren VERIFICACIONES. Las cosas que no conocemos requieren PRUEBAS. Estos enfoques se pueden dividir de nuevo en muchos enfoques de prueba / técnicas u opciones operacionales.

1.4 Definición de Prueba Ágil

OA-1.6	K2	Comprender y explicar la definición de prueba y especialmente de prueba ágil

Han habido muchas definiciones de Prueba Ágil (C. Kaner, M. Bolton, J. Bach, L. Crispin, E. Hendrickson, J. Gregory y muchos más) y discusiones sobre estas definiciones. La mayoría de las definiciones se centran en las pruebas o los testers como un papel separado en el desarrollo de software. Se han añadido dos elementos muy importantes que faltan a esta definición: la responsabilidad del equipo en la calidad, así como los cuatro tipos de habilidades (habilidades de la gente, habilidades ágiles, habilidades de prueba, habilidades tecnológicas) que los testers deben poseer.

La definición consta de 3 partes: **por qué** hacemos lo que hacemos, **qué** hacemos y **quién** lo hace.

¿Por qué?:

- Ayudar al cliente en el proceso de revisión de su producto proporcionando la mayor información posible sobre el funcionamiento y los riesgos que conlleva el uso del producto.
- Ayudar al equipo en la producción de un producto cada vez mejor, que cumpla con las expectativas e intenciones del cliente.

¿Qué hacemos?:

- Desarrollar la conciencia de la calidad en el equipo ayudando activamente a mejorar continuamente la calidad.
- Realizar pruebas de los criterios que no están claramente establecidos o aún no están totalmente determinados mediante la experimentación e investigación con métodos de prueba como las pruebas exploratorias, revisiones por pares y la búsqueda de errores.
- Realizar verificaciones (automatizadas) basadas en criterios establecidos utilizando métodos de prueba como la prueba de unitarias, la prueba de integración, la prueba de seguridad, la prueba de rendimiento y la prueba de aceptación.

¿Quién?

- Personas que dominan las habilidades interpersonales, habilidades ágiles, habilidades de prueba, habilidades tecnológicas, que tienen la motivación y la credibilidad y son capaces de obtener, retener y mejorar estas habilidades.

Detalladamente "Quién":

- Motivación: un tester necesita estar motivado para aprender mejorando sus habilidades. Sin estar motivado no puede mejorar su equipo.
- Credibilidad: al entregar valor que no es directamente un producto visible (percibido en la calidad), un tester en un contexto ágil necesita credibilidad dentro y fuera del equipo para hacer un buen trabajo. Si no tiene credibilidad, está constantemente defendiendo lo que está haciendo en lugar de añadir valor.
- Habilidades interpersonales: en un equipo, un tester necesita muchas habilidades interpersonales para hacer un buen trabajo como tester en un contexto ágil. Tiene que ser capaz, por ejemplo, de dar una buena retroalimentación, hacer preguntas difíciles, pensar críticamente y describir la historia de la prueba. Tiene que ser capaz de comunicarse con las partes interesadas, las empresas, los clientes, los desarrolladores y las personas encargadas de las operaciones.
- Habilidades ágiles: un tester, en un contexto ágil, necesita la mentalidad ágil para mantener el enfoque en entregar valor para sus clientes. Esto significa que usted necesita entender la agilidad y el ciclo de vida del desarrollo de software y cualquier método utilizado para entregar el valor (Scrum, XP, Kanban, DevOps, ...).
- Habilidades de prueba: la información sobre el estado del producto debe estar disponible de forma instantánea y constante. Esto requiere una estrategia de prueba realmente diferente y diferentes métodos de prueba. Un tester en un contexto ágil necesita una amplia gama de métodos a su disposición para poder utilizar el más adecuado en el contexto específico del proyecto y del producto.
- Habilidades tecnológicas: un tester en un contexto ágil necesita conocer la tecnología del producto y el entorno de desarrollo. Necesita ser capaz de leer código, escribir pruebas automatizadas en diferentes niveles y definir riesgos basados en la tecnología utilizada.

Representación visual elaborada por Karen Greaves y Sam Laing **[GA1]**

Gráfico 2

1.5 Scrum

OA-1.7	K2	Comprender el marco de Scrum
OP-1.1	OP-3	Compartir las definiciones actuales de Scrum entre el grupo
OA-1.8	K2	Explicar el marco de Scrum y el papel de un tester en el equipo
OA-1.9	K1	Recordar los roles del producto owner, del scrum máster y el equipo de scrum
OA-1.10	K1	Recordar el uso de la estimación de la planificación (como los puntos de historia) y los puntos de prueba en lugar de incluir estimaciones de prueba
OA-1.11	K2	Explicar la perspectiva de la prueba en la planificación del poker y el papel de la prueba en la planificación del sprint
OA-1-12	K1	Recordar la velocidad y el compromiso impulsados por la planificación
OA-1.13	K2	Comprender las estimaciones de las pruebas
OA-1.14	K2	Explicar el papel de la prueba en las sesiones de refinamiento
OA-1.15	K2	Explicar el papel de la prueba en las retrospectivas
OA-1.16	K2	Comprender el papel de la prueba en un programa de aumento / planificación de liberación
OA-1.17	K1	Aprender algunos anti-patrones scrum comúnmente usados como la iteración de prueba N+1 o las hardening iterations para resolver la deuda técnica

Hay muchos marcos ágiles: XP, Scrum, Kanban, Lean, Crystal y muchos más. Scrum se utiliza durante la formación como el marco más utilizado.

Hay muchas implementaciones y variaciones diferentes del marco ágil más comúnmente utilizado: Scrum. La gente suele empezar a hacer cambios en el marco original incluso antes de empezar a usarlo. Esto a menudo resulta en formas de trabajo no ágiles y no debería llamarse " ágil". Esta información es útil y necesaria para comprender el rol de un tester en un equipo ágil.

Scrum se basa en la teoría empírica de control de procesos, o empirismo. El empirismo afirma que el conocimiento proviene de la experiencia y la toma de decisiones basadas en lo que se conoce. Scrum

emplea un enfoque iterativo e incremental para optimizar la predicción y controlar el riesgo. Tres pilares sostienen cada implementación del control empírico de procesos: transparencia, inspección y adaptación **[SC1]**.

Se recuerdan los diferentes papeles en un equipo típico de scrum: propietario del producto, scrum máster y miembro del equipo y sus responsabilidades.

Se recuerdan los diferentes eventos en Scrum: scrum diario, refinamiento del registro de trabajo, planificación de sprint, revisión de sprint, retrospectiva.

Se examina más de cerca la estimación de trabajo según las sesiones de planificación de sprint. Se explica el papel del método de estimación más comúnmente usado, usando la estimación por puntos de historia versus la estimación por horas. También se discute la opción de tener puntos de prueba y puntos de desarrollo separados y unirlos para obtener la estimación, así como la opción de incluir los puntos de prueba sin mencionarlos explícitamente. Al dividir las funciones y los puntos la estimación parece más fácil; sin embargo, crea menos oportunidades de aprender unos de otros y de discutir sobre la historia (que es el objetivo de hacer la estimación). El papel del miembro del equipo con capacidad para realizar pruebas en la planificación de sprints también consiste en desafiar al equipo y mantenerlo alerta, así como hacer preguntas críticas sobre las historias. Otro papel de esto es hacer que el equipo comprenda que hacer un pequeño cambio en el programa a veces puede tener grandes consecuencias para las pruebas y las estimaciones de las pruebas. También mencionamos 2 formas comúnmente usadas para planificar un sprint: impulsado por la velocidad e impulsado por el compromiso. Difieren ligeramente; sin embargo, la planificación de un sprint se basa principalmente en el consenso, y no en la planificación.

La mentalidad de un tester es realmente valiosa en los refinamientos de requisitos para traer el aspecto del pensamiento crítico y un enfoque diferente de una historia (lo que potencialmente podría salir mal frente a cómo podemos hacer que funcione). Si en una organización no es una práctica común involucrar a un tester en los refinamientos, se deben tomar medidas para convencer al equipo de que involucre a todos los miembros del equipo incluyendo al tester en el refinamiento.

La retrospectiva es un evento muy importante en Scrum. Trae la posibilidad de reflexionar y aprender. Esto también significa que un equipo necesitará un entorno a prueba de fallos ya que los miembros del equipo comparten las cosas que probablemente fallaron para aprender de ello y hacerlo mejor la próxima vez. Esto significa que la gente necesitará aceptar el hecho de que pueden tomar una posición vulnerable y la mayoría de la gente lo encuentra difícil. Como testers, no tenemos un papel especial en una retrospectiva; sin embargo, podemos ayudar a nuestro equipo tomando primero la posición vulnerable y compartiendo un experimento (prueba) que realizamos y que falló y lo que aprendimos de ello.

Si un equipo se encuentra en un entorno de escala ágil, habrá sesiones de Incremento de Producto en las que un tester puede añadir mucho valor desde la perspectiva de la prueba hacia el resultado de las actividades y asegurarse de que la preparación organizacional está ahí desde la perspectiva de la prueba.

La parte de scrum está compuesta con algunas prácticas comúnmente usadas que se consideran anti patrones de ágil y scrum:

- La iteración N+1 o iteración de prueba retrasada: en la iteración N, se realiza el desarrollo y la prueba de ese desarrollo se completa en la iteración N+1. Las correcciones se hacen más probablemente en la iteración N+2.

- “hardening iterations” (Iteraciones de endurecimiento), como resultado de la acumulación de deuda técnica: las hardening iterations se utilizan para arreglar soluciones construidas de baja calidad (deuda técnica) en iteraciones anteriores. Si la refactorización forma parte de cada iteración de sprint, no hay necesidad de usar hardening iterations.
- Mini-cascada: en lugar de un scrum real, el equipo diseña, construye y prueba durante una iteración según el escenario de longitud en la cascada.

1.6 El tester en un contexto ágil

OA-1.18	K2	Comprender el cambio del papel del tester en un contexto ágil
OA-1.19	K2	Ser capaz de explicar cuál es el papel de los testers en un equipo ágil
OA-1.20	K2	Comprender el papel de un tester como el conductor de calidad clave en un equipo ágil
OA-1.21	K2	Comprender las habilidades que son esenciales para un tester en un contexto ágil

Se explican las diferencias entre el papel de un tester en un entorno tradicional y el papel de un tester en un contexto ágil. La función ha pasado de ser un tester funcional de requerimientos (que realiza principalmente verificaciones) a un "ingeniero de calidad", que en parte requiere diferentes habilidades (como se ha mencionado anteriormente).

En ágil, la calidad es una responsabilidad del equipo y si el equipo no asume esa responsabilidad, o no la entiende, el tester es responsable de ayudar al equipo a entender e implementar esta responsabilidad. El tester en un contexto ágil debe ser la persona que hace que el equipo sea consciente de la calidad. Ejemplos:

- Un tester en un contexto ágil motiva a los propietarios de productos/clientes a ser concisos sobre lo que quieren.
- Un tester en un contexto ágil se empareja con el propietario del producto/interesados desde el principio para que las historias de los usuarios sean lo más específicas posible.
- Un tester en un contexto ágil entrena a los desarrolladores sobre el valor de las buenas prácticas de codificación, incluyendo la prueba de unidad.
- Un tester en un contexto ágil es un conductor/arquitecto de calidad en el equipo ya que tiene la mentalidad de calidad de forma natural.

Para ser el ingeniero de calidad, un tester en un contexto ágil necesita habilidades específicas, tales como habilidades de trato con las personas/comunicación, para llevar mensajes/nuevas ideas al equipo y habilidades tecnológicas para ayudar a los desarrolladores con la creación de buenas pruebas unitarias y habilidades de prueba para explorar lo desconocido de la aplicación y muchas más habilidades de las cuales las más importantes se practican en esta formación.

1.7 Pensamiento crítico

OP-1.2	OP-3	Obtener las definiciones de los estudiantes para iniciar una discusión
OA-1.23	K2	Demostrar la definición de pensamiento crítico
OA-1.24	K2	Demostrar la definición de una reclamación o afirmación
OP-1.3	OP-1	Practicar con las reclamaciones, conclusiones y construcciones para comprender la complejidad
OA-1.25	K2	Comprender y reconocer las falacias
OA-1.26	K2	Comprender los errores de pensamiento y cómo influyen en el juicio de las personas
OA-1.27	K3	Aplicar todas las definiciones que juegan un papel en el pensamiento crítico
OP-1.4	OP-2	Practicar el pensamiento crítico usando argumentos, reclamos, construcciones, falacias y errores de pensamiento

El pensamiento crítico o escepticismo es generalmente una actitud de cuestionamiento o duda hacia uno o más elementos de conocimiento o creencia putativa o dogma. A menudo se dirige a dominios como el sobrenatural, la moral (escepticismo moral), el teísmo (escepticismo sobre la existencia de Dios) o el conocimiento (escepticismo sobre la posibilidad de conocimiento o de certeza).

Las afirmaciones y aserciones se están discutiendo como una declaración significativa, es decir:

- Verdadero
- No verdadero
- Ni "verdadero" ni "no verdadero"

La argumentación y el razonamiento se están discutiendo como una construcción de afirmaciones que causan una conclusión.

Una conclusión es válida si la negación de la conclusión está en contradicción con una de las afirmaciones utilizadas en la construcción. Un ejemplo:

- Afirmación 1: "Juan es un analista de pruebas".
- Afirmación 2: "Todos los analistas de pruebas son pensadores críticos".
- Conclusión: “Juan es un pensador crítico"

La conclusión es válida, ya que no puede negarla basándose en las afirmaciones 1 y 2. Sin embargo, la conclusión no es verdadera porque al menos 1 afirmación no es ni "verdadera" ni "no verdadera".

Las falacias se definen y demuestran cómo la gente a menudo utiliza las falacias para convencer a otras personas de que hagan lo que alguien quiere que hagan en lugar de hacer lo correcto. Un tester necesita aprender a reconocer las falacias ya que juegan un papel importante en ser crítico. Ser crítico es uno de los valores más importantes que un tester puede aportar al equipo.

La gente sigue considerando que los errores **[KM1]** surgen de un juicio irracional. Las conclusiones se sacan de forma irracional. La gente continúa en el camino del pensamiento crítico, aunque es un viaje doloroso y difícil. Se entiende por qué no ser crítico parece mucho más fácil (aunque al final no lo es).

El viaje termina dando múltiples ejemplos de cómo convertirse en un pensador crítico y practicar el ser crítico con un ejercicio usando un juego serio.

1.8 Retrospectivas

OA-1.28	K2	Comprender la definición de una retrospectiva
OA-1.29	K2	Explicar el propósito de las retrospectivas
OA-1.30	K2	Demostrar la necesidad de las retrospectivas y el papel de un tester
OA-1.31	K3	Ser capaz de implementar retrospectivas en un equipo
OA-1.32	K1	Poder seleccionar diferentes tipos de retrospectivas
OP-1.5	OP-3	¿Cómo puede un juego serio ser usado para una retrospectiva?

La retrospectiva es un evento muy importante en scrum. Trae la posibilidad de reflexionar y aprender. Esto también significa que un equipo necesitará un entorno a prueba de fallas ya que los miembros del equipo comparten las cosas en las que probablemente fallaron para aprender de ellas y hacerlo mejor la próxima vez. Esto significa que la gente necesita la confianza de que pueden tomar una posición vulnerable y la mayoría de la gente lo encuentra difícil. Como testers, no tenemos un papel especial en una retrospectiva; podemos ayudar a nuestro equipo, sin embargo, tomando primero la posición vulnerable y compartiendo un experimento (prueba) que realizamos y que falló y lo que aprendimos de ello.

El aprendizaje continuo juega un papel clave en ágil. Como tester, necesitas entender el papel de una retrospectiva y ser capaz de realizar una retrospectiva también.

La definición de una retrospectiva según la guía scrum es: "La Retrospectiva de sprint es una oportunidad para que el equipo de scrum se inspeccione a sí mismo y cree un plan de mejoras que se llevará a cabo durante el próximo sprint".

La definición se discute en detalle para explorar el significado real de los puntos más destacados de esta definición:

- Oportunidad
- El equipo scrum se inspecciona a sí mismo
- Crear un plan

Al discutir el significado de las retrospectivas, llegamos a su verdadero propósito: ser críticos con uno mismo y con los demás para aprender y hacerlo mejor la próxima vez (mejora continua).

Se muestra una variedad de diferentes retrospectivas con el fin de producir diferentes tipos de enfoque para mejorar - podría ser el trabajo en equipo, la calidad, las herramientas, la comunicación, etc. - y para mantener las retrospectivas interesantes.

Capítulo 2 - Empezar a probar - Un caso

La formación práctica se construye alrededor de un caso que se utiliza para explicar la teoría. El caso contiene software real (y hardware opcional) **[SH1]** para ser probado.

Palabras clave

Ágil, pruebas, estrategia de pruebas, preguntas, retroalimentación, pensamiento crítico, riesgos, riskstorm(tormenta de riesgo), pruebas exploratorias, juego de encabezados, automatización de pruebas, verificación de automatización, oráculos, heurística, mnemotecnia, estructura del producto, estructura del proyecto, temas.

OP-2.1	OP-3	Empezar a probar software desconocido que opcionalmente se ejecuta en un hardware específico
OA-2.1	K3	Optimizar el uso de la situación dada, la documentación y otra información disponible
OA-2.2	K1	Recordar la necesidad de hacer preguntas antes de hacer algo para entender la situación específica
OA-2.3	K1	Recordar la necesidad de hacer preguntas antes de hacer algo para entender las necesidades específicas
OA-2.4	K1	Recordar la necesidad de hacer preguntas antes de hacer algo para comenzar la prueba en una dirección útil
OP-2.2	OP-3	Probar un objeto específico (algo que se parece a una pelota)
OA-2.5	K2	Comprender la necesidad de hacer preguntas
OA-2.6	K2	Comprender el valor de las preguntas en general
OA-2.7	K2	Comprender el valor de las preguntas en relación con las pruebas
OA-2.8	K2	Comprender la definición de retroalimentación
OA-2.9	K2	Comprender el significado y el valor de la retroalimentación
OA-2.10	K2	Replantear la retroalimentación en las pruebas
OA-2.11	K2	Comprender las reglas básicas de la retroalimentación
OA-2.12	K3	Aplicar 4 métodos diferentes de retroalimentación
OP-2.3	OP-2	Practicar la retroalimentación en una situación de prueba en la vida real
OA-2.14	K2	Entender qué son los modelos y cómo usamos los modelos (mental, conceptual y cosas reales/mundo)
OA-2.15	K3	Aplicar modelos para el desarrollo de software y pruebas en especifico
OP-2.4	OP-3	Crear un esquema de proyecto y presentarlo a los demás
OA-2.16	K2	Comprender la definición de un esquema de proyecto
OA-2.17	K3	Aplicar el esquema del proyecto a su proyecto
OA-2.18	K2	Comprender la relación entre el esquema de un proyecto y las pruebas
OA-2.19	K1	Recordar que hay que hacer preguntas para obtener la información necesaria para el esquema del proyecto
OA-2.20	K3	Aplicar haciendo preguntas para obtener un buen esquema del proyecto
OA-2.21	K3	Utilizar el mnemotécnico PROCREA para ayudar a crear el esquema del proyecto
OP-2.5	OP-3	Crear un esquema del producto explorando el producto (Caso software/hardware)
OA-2.22	K2	Comprender la definición de un esquema de producto
OA-2.23	K3	Aplique el esquema del producto a su proyecto
OA-2.24	K2	Comprender la relación entre el esquema de un producto y las pruebas

OA-2.25	K1	Recordar hacer preguntas para obtener la información necesaria para el esquema del producto
OA-2.26	K3	Aplicar haciendo preguntas para obtener un buen esquema del producto
OA-2.27	K3	Utilizar PRODUCTO para ayudarnos a crear el esquema del producto

2.1 Prueba del software

OP-2.1	OP-3	Empezar a probar software desconocido que opcionalmente se ejecuta en un hardware específico
OA-2.1	K3	Optimizar el uso de la situación dada, la documentación y otra información disponible

Los proyectos de la vida real suelen estar en marcha cuando un nuevo miembro del equipo se involucra. Como nuevo miembro tienes que tratar con software y entornos ya existentes, y normalmente hay poco tiempo y no hay documentación existente sobre cómo sesupone que funciona el software, o bien, la documentación ya es obsoleta.

Por medio de una situación real de prueba del proyecto, usted, como tester, aprende cómo comenzar sus pruebas utilizando la información disponible y haciendo preguntas para obtener la información que requiere/necesita.

2.2 Preguntas

OA-2.2	K1	Recordar la necesidad de hacer preguntas antes de hacer algo para entender la situación específica
OA-2.3	K1	Recordar la necesidad de hacer preguntas antes de hacer algo para entender las necesidades específicas
OA-2.4	K1	Recordar la necesidad de hacer preguntas antes de hacer algo para comenzar la prueba en una dirección útil
OP-2.2	OP-3	Pruebe un objeto específico (algo que se parece a una pelota)
OA-2.5	K2	Comprender la necesidad de hacer preguntas
OA-2.6	K2	Comprender el valor de las preguntas en general
OA-2.7	K2	Comprender el valor de las preguntas en relación con las pruebas

Antes de probar algo, el tester necesita obtener tanta información como sea posible:

- ¿Quién le pide que haga la prueba y cuál es su relación con el tema?
- ¿Cuá crees que sea el tema y qué significa eso para ti?
- ¿Cuál es el uso previsto?
- ¿Qué deberías probar, qué es importante, dónde están los posibles riesgos?
- ¿Se ha probado ya antes y hay resultados de estas pruebas disponibles?
- ¿Qué se ha cambiado recientemente?
- ¿De cuánto tiempo se dispone?
- ¿Quién está o estuvo involucrado y puede ser consultado para obtener más información?

- Se pueden hacer muchas más preguntas según el contexto y la información disponible

Se explica la necesidad de entender por qué, quién y cuándo se deben hacer las preguntas:

- Para obtener más información sobre cualquier cosa, en cualquier momento.
- Cuando las cosas no están claras.
- Lo que crees que es verdad no tiene por qué serlo.
- Lo que sabes no esta completo.
- Para comprobar si entiendes los riesgos reales.

2.3 Retroalimentación

OA-2.8	K2	Comprender la definición de retroalimentación
OA-2.9	K2	Comprender el significado y el valor de la retroalimentación
OA-2.10	K2	Replantear la retroalimentación de las pruebas
OA-2.11	K2	Comprender las reglas básicas de la retroalimentación
OA-2.12	K3	Aplicar 4 diferentes tipos de retroalimentación
OP-2.3	OP-2	Practicar la retroalimentación en una situación de prueba en la vida real

La definición de retroalimentación es: "Reaccionar ante un comportamiento cambiante o productos y procesos cambiantes”.

Un tester proporciona una retroalimentación que es el resultado más importante del trabajo realizado. El reto es que esta retroalimentación no suele ser una retroalimentación positiva. Por lo general, los testers encuentran problemas en la calidad de lo que otra persona ha creado. La retroalimentación no positiva puede sentirse negativa hacia el creador del producto.

Por esa razón, es importante que un tester pueda dar una retroalimentación de la manera más constructiva, a fin de obtener los mejores resultados. Se dan las reglas básicas de la retroalimentación y se proporcionan 4 formas diferentes de dar retroalimentación para que pueda elegir la forma que, en su situación específica, tenga el mayor grado de éxito:

- Retroalimentación correctiva; comportamiento, sentimientos, consecuencias, comportamiento deseado.
- Evaluación de la retroalimentación; el método Pendleton **[PD1]**.
- Activando la retroalimentación; el método de la hamburguesa **[HB1]**.
- Retroalimentación motivadora; un cumplido.

Se practican 4 formas de proporcionar y recibir retroalimentación.

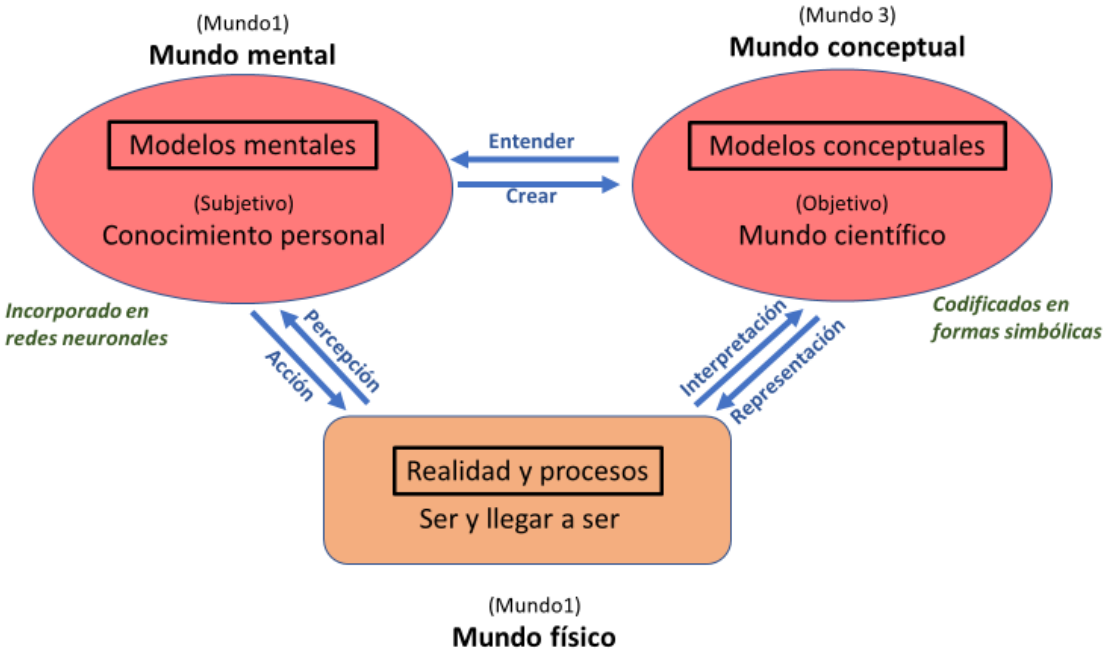
2.4 Modelos

OA-2.14	K2	Entender qué son los modelos y cómo la gente usa los modelos (mentales, conceptuales y cosas reales/mundo)
OA-2.15	K3	Aplicar modelos para el desarrollo de software y las pruebas

Los modelos son una simplificación del mundo real. Se pueden hacer modelos para todo lo que nos rodea; un nuevo producto como un coche o una casa, pero también modelos de un evento que se organiza, como una fiesta de cumpleaños.

La gente usa y necesita modelos porque el mundo real es demasiado complicado. Sólo trate de imaginar una fiesta de cumpleaños y escriba todo lo relacionado con esa fiesta de cumpleaños. Probablemente se le ocurra la gente, el pastel, las bebidas, etc., sin embargo, nunca estará seguro de quién asistirá realmente hasta que esté realmente allí. Tampoco sabe con seguridad qué quieren beber...

Así que la gente usa modelos para lidiar con situaciones que son demasiado complicadas de manejar. La creación de software y también la prueba de software es una de esas actividades. Es imposible tratar con la complejidad sin usar todo tipo de modelos (Mental, Conceptual y Cosas Reales/Mundo) para poner algo de orden en el caos. Algunos ejemplos de modelos son, por ejemplo, los modelos de una base de datos y los modelos de usuario.



Fuente: David Hestenes **[DH1]**

Los testers (la gente en general) tienen un amplio conjunto de modelos preconcebidos en su mente. Para el desarrollo y prueba de software, podemos mencionar modelos mentales, modelos técnicos, modelos de dominio y en general modelos de experiencia.

Al probar el software, los testers vinculan lo que ven (en el producto o proyecto o documentación) a sus propios modelos y sacan conclusiones lo. Cuanto más profundamente conectamos el modelo con lo que vemos, más nos centramos. Sin embargo, la gente no puede estar enfocada todo el tiempo, y el enfoque también reduce la escala de lo que se puede ver, así que tenemos que ser conscientes de que también tenemos que desenfocarnos para ver el panorama general.

2.5 Panorama general de la estrategia de prueba

Para crear las mejores perspectivas en el estado del producto, el equipo necesita un enfoque sobre la calidad. En el plan de estudios esto se conoce como la estrategia de prueba. La estrategia de prueba consiste en diferentes partes de información:

- Un esquema de proyecto para obtener y compartir información sobre el medio ambiente.
- Un esquema de producto para obtener y compartir información sobre el producto mismo.
- Un esquema de riesgo para obtener y compartir información sobre los posibles riesgos.
- Un esquema de prueba para obtener y compartir información sobre el enfoque de prueba elegido:
    - Qué tipo de pruebas se realizan.
    - Qué pruebas se hacen y cuándo.
    - Entornos de prueba y herramientas.
    - Reporte.

Tengan en cuenta que una estrategia de prueba es un producto incremental. Se desarrolla con el tiempo. No es necesario que esté completa desde el principio. También tenga en cuenta que todas las partes de una estrategia de prueba no deben ser un duplicado de los productos ya existentes. En ese caso, Incluya una referencia.

El propósito de una estrategia de prueba no es sólo obtener información, sino también compartir y discutir cualquier tema relacionado con la prueba dentro (y potencialmente también fuera) del equipo. Una estrategia de prueba es una reflexión de la vida real sobre el proyecto, el producto, el estado del producto, los riesgos y la forma en que el equipo trata la calidad y las pruebas.

Es muy recomendable visualizar la estrategia de la prueba para mantenerla viva y recordar constantemente al equipo la necesidad de la calidad y ser consciente de los elementos que pueden influir en ésta.

2.6 Esquema del proyecto

OP-2.4	OP-3	Crear un esquema del proyecto y presentarlo a los interesados
OA-2.16	K2	Comprender la definición de un esquema de proyecto
OA-2.17	K3	Aplicar el esquema del proyecto a su proyecto
OA-2.18	K2	Comprender la relación entre el esquema de un proyecto y las pruebas
OA-2.19	K1	Recordar que hay que hacer preguntas para obtener la información necesaria para el esquema del proyecto
OA-2.20	K3	Presentar preguntas para obtener un buen esquema del proyecto
OA-2.21	K3	Utilizar el PROYECTO mnemotécnico para ayudarnos a crear el esquema del proyecto

Cada proyecto y cada cosa en su entorno puede tener un impacto en la decisión de probar o no probar algo. Como tester en un contexto ágil, es necesario ser capaz de crear un esquema de proyecto para obtener información sobre el proyecto para tomar decisiones, junto con el equipo, sobre qué probar o qué no es necesario probar. Mucha de esta información está escondida en la cultura, la gente, la historia, los clientes, el conocimiento implícito y así sucesivamente. Los testers

deben ser capaces de hacer las preguntas correctas para aclarar esta información para que podamos desplegar la estrategia y el enfoque correcto en las pruebas.

El mnemotécnico PROCREA mnemotécnico le ayuda a crear un esquema de proyecto y contiene la siguiente información:

Propósito	Misión y visión
Relaciones	Equipo de scrum / Equipo de desarrollo
Objetivos	Indicadores clave de rendimiento
Clientes	Interesados, Patrocinadores, Grupos
Riesgo	Las posibilidades del proyecto y las amenazas al mismo
Entorno	Entorno técnico (herramientas, hardware, software, información (requisitos, etc.))
Adaptación	Reglas del equipo, valores del equipo, acuerdos de trabajo

Propósito

- Visión
  - ¿Cómo cambia el producto el mundo de los usuarios (por ejemplo, un trabajo menos repetitivo)?
  - ¿Qué problema se resuelve / qué beneficio se proporciona, y para quién?
- Misión
  - Define los usuarios del producto a crear.
  - Define las acciones y los resultados finales del equipo.
  - Define el producto o servicio que el equipo entregará.
  - Define los atributos del producto o servicio que describe el valor añadido.
  - Define el valor genérico del producto para el cliente.

Relaciones

- ¿Qué aporta cada miembro del equipo?
- ¿Qué hace feliz a cada miembro del equipo?

Objetivos

- KPIs (por sus siglas en Ingles, Indicadores clave de rendimiento) que están claros para el equipo
  - Externo: el 1 de julio de 2020 (después de la iteración 5) el nuevo software reducirá la ocupación del mostrador de atención al cliente en un 20% en dos meses.
  - Interno: el 1 de marzo de 2020, la producción del equipo aumentó en un 10% (N puntos de historia por iteración de 100 a 110).

Clientes

- Interesados: el cliente: ¿cuál es su interés?
- Patrocinadores: ¿quién en este proyecto quiere que se realicen pruebas? ¿Es alguien distinto de la parte interesada?
- Equipos: ¿Hay otros equipos o su propio equipo, tal vez clientes también?

Riesgo

- Aunque todo el "proyecto" sólo lleva de 2 a 3 días, también se pueden analizar las oportunidades y los riesgos en un período corto:
  - ¿Está el product owner / cliente / patrocinador disponible durante esos 3 días??
  - ¿Sólo hay un objeto de prueba? ¿Y si éste se daña??
  - ¿Debemos centrarnos en la cancelación del acuerdo para el producto, para que podamos terminar mucho antes (y ¿queremos eso?)?

Entorno

- Instrucciones.
- El software.

- Cualquier hardware específico.
- Cualquier herramienta que puedas usar o tengas que usar.

Adaptación

- Reglas del equipo
  - Ejemplos
    - Apertura: "Habla sobre temas que te conciernen a ti o al equipo"
    - Coraje: "Comunícate abiertamente y ponlo todo sobre la mesa."
  - Acuerdos de trabajo
  - Ejemplos
    - Durante las reuniones, nos apegamos al tema.
    - Determinamos en este orden: primero los expertos, sociocrático (sin objeciones), consenso (todos a favor), democrático.
    - Aprendizaje: fallamos rápido, mucho e identificamos nuestros errores lo antes posible.
    - Respetamos nuestros plazos. En caso de fuerza mayor, informaremos a todo el equipo y a los interesados.

2.7 Esquema del producto

OP-2.5	OP-3	Crear un esquema del producto explorando el producto (software/hardware)
OA-2.22	K2	Comprender la definición de un esquema de producto
OA-2.23	K3	Aplicar el esquema del producto a su proyecto
OA-2.24	K2	Comprender la relación entre el esquema de un producto y las pruebas
OA-2.25	K1	Recordar hacer preguntas para obtener la información necesaria para el esquema del producto
OA-2.26	K3	Aplicar haciendo preguntas para obtener un buen esquema del producto
OA-2.27	K3	Utilizar el acrónimo PRODUCT para ayudarnos a crear el esquema del producto

Para entender lo que estamos probando, necesitamos una visión general del producto en sí. Un tester necesita crear un mapa que le ayude a entender el producto. Si no entendemos el producto, no podemos probarlo adecuadamente.

Un mnemotécnico útil para ayudarnos es PRODUCT:

Plataforma Con qué funciona el producto (técnicamente)
Relaciones ¿Cómo? nosotros (u otro sistema) interactuamos con el sistema
Operación ¿Cómo funciona el producto?
Datos Lo que el producto procesa
Usuarios Todos los elementos del producto
Construcción Cómo se utiliza el producto
Tiempo Relaciones entre el producto y el tiempo

Plataforma

- Hardware
- Middleware
- Software

Relaciones

- Interfaces de usuario
- Interfaces del sistema

- Interfaces API

Importación/Exportación

Operación

- Aplicación: cualquier función que defina el producto.
- Cálculo: cualquier función matemática en el producto.
- Seguridad: derechos de usuario, seguridad de los datos, encriptación, seguridad del "front y back end", vulnerabilidades en los subsistemas.
- Transformaciones: configuración de fuentes, inserción de clip art, transferencias de dinero, etc.
- Y más.

Datos

- ¿Cómo maneja el producto los datos?
    - Entrada / salida
    - Persistente
    - Inválido / Ruido
    - Ciclo de vida (CRUD)
    - Y más

Usuarios

- Usuarios
- Ambiente (físico, luz, distracciones, ruido)
- Uso común (patrones, secuencias o entrada)
- Uso desfavorable (errores, ignorancia, uso malicioso, uso indebido)
- Uso extremo

Construcción

- Código: de los ejecutables a las rutinas individuales.
- Hardware: cada componente de hardware integral para el proyecto.
- No ejecutables: todos los archivos que no sean multimedia o programas, como datos de muestra, archivos de ayuda.
- Colateral: todo lo que va más allá de lo anterior: enlaces y contenidos web, empaquetado, licencias, etc.

Tiempo

- Entrada, salida, retrasos e intervalos.
- Rápido / Lento: entrada
- Porcentajes de cambio: picos, arranques, caídas, cuellos de botella, interrupciones.
- Concurrencia: multiusuario, tiempo compartido, Interacciones, datos compartidos.
- Ajustes de tiempo de espera, periodos, zonas horarias, vacaciones de la empresa, periodos de garantía, funciones de cronometraje.

Capítulo 3 – Riesgos

"Riesgo" es la palabra clave para cualquier estrategia de prueba. Sin embargo, en un entorno ágil, los riesgos cambian y provienen de fuentes más diferentes que antes. Esto requiere una estrategia diferente para tratar los riesgos.

Palabras clave

Ágil, Prueba, Estrategia de prueba, Pensamiento crítico, Riesgos, “Riskstorming”, “headline game”.

OA-3.1	K2	Comprender lo que es importante en un producto y lo que no en relación con los riesgos
OA-3.2	K2	Comprender que un riesgo sólo es valioso si es lo suficientemente específico
OA-3.3	K2	Comprender la definición de un riesgo
OA-3.4	K2	Comprender los desafíos cuando se trata de definir los riesgos
OA-3.5	K2	Ser capaz de explicar los diferentes tipos de riesgos
OA-3.6	K2	Poder separar los diferentes tipos de riesgos
OA-3.7	K2	Comprender cómo el riesgo de tipo A puede influir en los riesgos de tipo B
OA-3.8	K2	Comprender el papel de un tester en relación con los riesgos y el producto
OA-3.9	K2	Comprender lo que es un “oráculo”
OA-3.10	K2	Estar atento a los oráculos
OA-3.11	K1	Comprender el valor de los oráculos en las pruebas
OA-3.12	K1	Comprender cual es el riesgo de los oráculos basados en principios o mecanismos inválidos
OA-3.13	K2	Comprender los dos enfoques diferentes del análisis de riesgos
OA-3.14	K3	Aplicar el mejor enfoque (combinación también) a su situación
OP-3.1	OP-2	Realizar un análisis de riesgo en el caso del producto basado en los métodos dados
OP-3.2	OP-2	Realizar la actividad “headline game” de riesgo en el caso del producto
OP-3.3	OP-2	Presentar su análisis de riesgo al grupo basado en la actividad “headline game”.
OP-3.15	K3	Ser capaz de ejecutar un “nightmare headline”.
OP-3.4	OP-2	Ejecutar un “Testsphere riskstorm” con instrucciones del formador
OP-3.5	OP-2	Presente su análisis del “riskstorm” al grupo
OA-3.16	K1	Recordemos el diferente enfoque entre el ágil y cascada
OA-3.17	K1	Relacionar ágil con la evaluación de riesgos
OA-3.18	K2	Demostrar el ciclo continuo de evaluación de riesgos

3.1. Definición de riesgo

OA-3.1	K2	Comprender lo que es importante en un producto y lo que no en relación con los riesgos
OA-3.2	K2	Comprender que un riesgo sólo es valioso si es lo suficientemente específico
OA-3.3	K2	Comprender la definición de un riesgo
OA-3.4	K2	Comprender los desafíos cuando se trata de definir los riesgos

Para identificar los riesgos, un equipo necesita saber lo que es importante y lo que no. Algo que no es importante no es un riesgo.

Cualquier cosa que sea importante puede ser un riesgo, sin embargo, no todo lo importante puede o será o debe ser probado para mitigar un riesgo.

Un riesgo sólo es valioso si es lo suficientemente específico. Por ejemplo: es un riesgo si la aplicación es demasiado lenta. Esto es un riesgo; sin embargo, no es muy útil para las pruebas. ¿Qué es "demasiado lenta": es toda la aplicación o sólo partes específicas?

Esto significa que hay algunas preguntas difíciles de hacer mientras se prueba un riesgo:

- ¿Qué es una amenaza para el valor del producto?
- ¿Quiénes son las personas que importan?
- ¿Cómo determinamos la probabilidad real?
- ¿Cómo determinamos su impacto?

Para poder hablar el mismo lenguaje en un equipo, necesitamos empezar con una definición clara de la terminología utilizada. Se utiliza la siguiente definición de un riesgo:

"Riesgo es cualquier cosa que amenaza el valor de un producto para una persona que importa" **[JW1]**.

El riesgo es una combinación de la probabilidad de que el riesgo se produzca y el impacto que tendrá sí se convierte en “realidad”.

3.2 Tipo y área de riesgos

OA-3.5	K2	Ser capaz de explicar los diferentes tipos de riesgos
OA-3.6	K2	Poder separar los diferentes tipos de riesgos
OA-3.7	K2	Comprender cómo los riesgos de los proyectos pueden influir en los riesgos de los productos

Hay diferentes tipos de riesgos:

- Riesgos del proyecto para los cuales podemos usar PROCREA (esquema del proyecto) como base.
- Riesgos del producto para los cuales podemos usar el PRODUCT (esquema del producto) como base.

Los riesgos pueden influir en otros riesgos, también a través de diferentes tipos de riesgos. Ejemplo: tener demasiados desarrolladores sin experiencia es un riesgo del proyecto. Es probable que este riesgo se convierta también en un riesgo del producto. Los desarrolladores inexpertos son más propensos a cometer errores debido a su falta de experiencia. Esto amenaza el valor del producto y probablemente requiere más esfuerzo en las pruebas.

3.3 Cobertura de riesgos

OA-3.8	K2	Comprender el papel de un tester en relación con los riesgos y el producto

Hay 3 estados importantes de un producto que se relacionan con los riesgos:

- Lo que ya sabemos.
- Sabemos todo lo que hay que saber.
- Sabemos lo suficiente para tomar una decisión informada sobre el producto.

Los equipos, y especialmente los testers, tienen que hacer preguntas, confiar en su intuición y explorar. ¿Por qué? Debido al espacio de RIESGO (RISK GAP): es tarea de cada miembro del equipo (y del tester que es el que dirige) asegurarse de que sabemos todo lo posible sobre el producto (y el proyecto) que se va a construir, de modo que podamos obtener suficiente información sobre los riesgos asociados. De modo que, en última instancia, sobre la base de las medidas de prueba resultantes, se pueda tomar una "decisión informada" sobre el seguimiento.

Gráfico 3

3.4 Oráculos

OA-3.9	K2	Comprender lo que es un oráculo
OA-3.10	K1	Estén atentos a los oráculos
OA-3.11	K1	Comprender el valor de los oráculos en las pruebas
OA-3.12	K1	Comprender cual es el riesgo de los oráculos basados en principios o mecanismos inválidos

Un oráculo es un principio o mecanismo por el cual las personas reconocen un problema **[BO1]**. En las pruebas, ese suele ser nuestro resultado esperado **[TE1]**.

Los oráculos son muy poderosos para los testers, si son conscientes de ellos, ya que les ayudan a reconocer los posibles problemas que amenazan el valor del producto: ¡los riesgos!

Los oráculos también pueden ser "peligrosos" si los principios o mecanismos utilizados son inválidos. Esto dará lugar a muchos "problemas" que no son un problema real.

3.5 De adentro hacia afuera vs. De afuera hacia adentro, en el análisis de riesgos

OA-3.13	K2	Comprender los dos enfoques diferentes del análisis de riesgos
OA-3.14	K3	Aplique el mejor enfoque (combinación también) a su situación

El análisis de riesgos puede ser abordado de dos maneras diferentes. Por lo general, ambas formas deben combinarse para ser completas en sus evaluaciones de riesgo:

- De adentro hacia afuera: comenzamos con el conocimiento disponible de las personas involucradas
- De afuera hacia adentro: partimos de atributos de calidad genéricos y listas de control de riesgo

De adentro hacia afuera:

Los expertos y las partes interesadas elaboran una lista de posibles vulnerabilidades a nivel de los componentes y de lo que podría salir mal en cada uno de ellos. También piensan en qué situación dada podría llevar a que el componente fallara y, finalmente, piensan en quiénes podrían ser las víctimas de este fallo.

Los equipos necesitan tener conocimientos (técnicos) específicos sobre el sistema para tomar la vía de adentro hacia afuera.

De afuera hacia adentro:

En lugar de optar por el nivel muy detallado, se mantiene el detalle a atributos de calidad mucho más genéricos y listas de riesgo genéricas o historias de usuarios. Esto resultará en una lista diferente de riesgos potenciales. Si (aún) no tiene conocimientos técnicos más específicos disponibles, este método es el punto de partida.

3.6 Análisis de riesgos: headline game

OP-3.1	OP-2	Realizar un análisis de riesgo en el caso del producto basado en los métodos dados
OP-3.2	OP-2	Implementar la actividad enfocada al riesgo "headline game" basada en el caso del producto.
OP-3.3	OP-2	Presentar su análisis de riesgo basado en la actividad "headline game"

Se realiza un análisis de riesgo de afuera hacia adentro basado en el caso utilizando el Juego de "headline game" **[EH1]**. Este método se basa en la definición de riesgos sobre lo que el equipo/organización no quiere que salga en los anuncios por la mañana sobre el software que el equipo acaba de lanzar ayer.

Paso 1: Lluvia de ideas sobre una lista de fallos graves
Paso 2: Elegir un riesgo en el que trabajar
Pedirle al grupo que busque en la lista una pesadilla que se destaque como:

- Plausible
- Relacionada con el software
- Interesante

Paso 3: Lluvia de ideas que contribuyen a las causas
Paso 4: Refinar las causas en los casos de prueba
Paso 5: Lavar, enjuagar y repetir

3.7 Análisis de riesgos: “riskstorm” utilizando “testsphere” ©

OP-3.4	OP-2	Ejecutar un “testsphere” “riskstorm” con indicaciones del formador
OP-3.5	OP-2	Presentar su análisis de riesgo de “riskstorm” al grupo

Después de la actividad "headline game", se utiliza una forma completamente diferente de determinar los riesgos mediante el uso de (de adentro hacia afuera) "riskstorm". Usamos las tarjetas de "Testsphere" **[TS1]**. Este juego serio te ayudará a identificar los riesgos. Introducimos las tarjetas azules (criterios de calidad) y las rosas (heurística) para determinar y posteriormente ampliar los riesgos.

Gráfico 4

3.8 Riesgos en ágil: el ciclo continuo de evaluación de riesgos

OA-3.16	K1	Recordemos el diferente enfoque entre el ágil y cascada
OA-3.17	K1	Relacionar ágil con la evaluación de riesgos

OA-3.18	K2	Demostrar el ciclo continuo de evaluación de riesgos

En un entorno ágil, la evaluación de los riesgos es un proceso continuo. Los equipos no pueden hacer sólo un análisis de riesgo del producto y utilizarlo en todo el proyecto. Cada vez que creen o modifiquen o prueben algo, obtendrán nueva información sobre el producto y sobre los riesgos potenciales. Estos son los riesgos reales. Basándose en los resultados, el equipo puede enviar el producto o hacer algunos arreglos y/o hacer algunas pruebas más. Este ciclo funcionará para siempre mientras el equipo esté entregando el software.

El nivel en el que el equipo pasa por el ciclo suele coincidir con el ciclo de entrega del producto a producción. Sin embargo, cualquier tiempo del ciclo está bien: desde el lanzamiento o la iteración hasta la historia o el reporte etc.

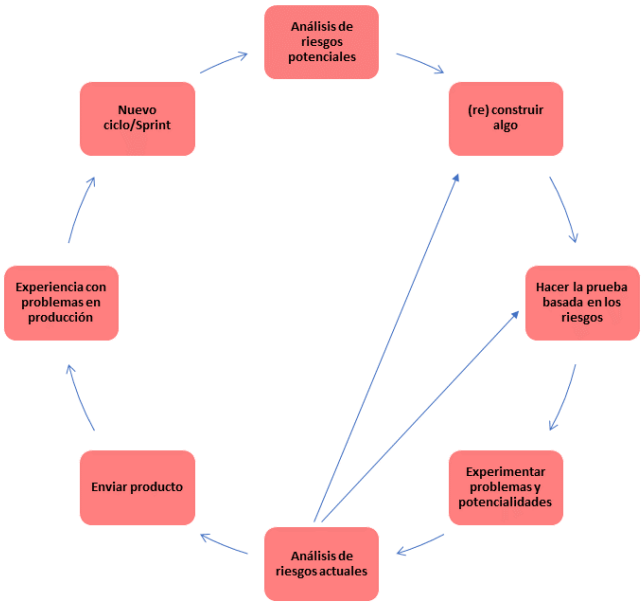


Gráfico 5

- Nuevo ciclo/iteración: inicio del (nuevo) ciclo. El nivel puede ser cualquiera (liberación, iteración, característica, etc).
- Analizar los riesgos potenciales: utilice una herramienta de análisis de riesgos de su elección para obtener una visión de los riesgos potenciales.
- (Re)construya algo: los miembros del equipo con un rol de desarrollador crearán el software.
- Tan pronto como sea posible, se entregará el nuevo componente para realizar pruebas (cualquier prueba que se ajuste a los riesgos asignados).
- Lo más probable es que el equipo descubra problemas (potenciales). Los problemas se arreglan y se prueban de nuevo. Durante las pruebas también se pueden encontrar nuevos riesgos que necesitan más pruebas.
- Cuando se cierra la brecha de riesgo (y cuando el DP/ interesados deciden) el equipo puede enviar el producto.
- Después del envío se pueden encontrar problemas (y riesgos) y el ciclo comienza de nuevo.

Capítulo 4 - Historias de usuarios

Las historias de usuario son la forma más común de reportar/documentar las especificaciones. Aunque las historias de usuario son menos complejas que un diseño completamente funcional y técnico, podrían crear problemas dentro del equipo ágil. Algunos ejemplos son: las historias de usuario pueden ser poco claras, incompletas, demasiado grandes, no comprobables o demasiado dependientes de otras historias.

Esto es una amenaza a la calidad del resultado final y por lo tanto hace que sea muy relevante para un tester conocer los detalles sobre las buenas historias de usuario.

Palabras clave

Valor, retroalimentación, formato (¿QUIEN?, ¿QUÉ?, ¿POR QUÉ?), específico, criterios de aceptación (Dados CUANDO formato, límite, consenso, base de prueba, base para la planificación), cuento de terror, división de historias, (Flujo de trabajo, CLAB (Crear, Leer, Actualizar, Borrar), Roles), PINVET (Pequeño, Independiente, Negociable, Valioso, Estimable, Testeable).

OA-4.1	K2	Comprender los valores de las historias de usuarios
OA-4.2	K1	Recordar los elementos de una historia de usuario
OA-4.3	K2	Demostrar el formato con un ejemplo
OA-4.4	K2	Comprender las complicaciones al usar el formato
OA-4.5	K1	Definir un criterio de aceptación
OA-4.6	K2	Comprender el papel de los testers en la definición y elaboración de los criterios de aceptación
OA-4.7	K2	Entender los escollos de la definición del criterio de aceptación
OA-4.8	K2	Entender los cuentos de terror como un medio para encontrar criterios de aceptación
OA-4.9	K2	Entender la división de la historia
OA-4.10	K2	Comprender cómo dividir las historias
OA-4.11	K1	Recordar cuándo dividir una historia
OA-4.12	K2	Entender cuando una historia está bien dividida
OP-4.1	OP-3	Analizar y rediseñar la historia dada

4.1 ¿Por qué usamos historias de usuario?

OA-4.1	K2	Comprender los valores de las historias de usuarios

Las historias de usuarios se utilizan por lo menos por dos razones: crean valor para el cliente y son un medio para entregar retroalimentación sobre las actividades de un equipo **[US1].**

4.2 Los elementos de una historia de usuario

OA-4.2	K1	Recordar los elementos de una historia de usuario

Explique que las historias de usuario consisten en la conversación real sobre las necesidades de los clientes y alguna forma de escritura con un título corto y único, un ¿QUIÉN?, ¿QUÉ Y POR QUÉ? **[AA1]** de la historia, y uno o más criterios de aceptación.

Según el concepto 3C **[JF1],** una historia de usuario es la conjunción de tres elementos:

- Tarjeta: La tarjeta es el medio físico que describe una historia de usuario. Identifica el requisito, su importancia crítica, el desarrollo esperado y la duración de la prueba, y los criterios de aceptación de esa historia. La descripción tiene que ser exacta, ya que se utilizará en el producto que se está investigando.
- Conversación: La conversación explica cómo se utilizará el software. La conversación puede ser documentada o verbal. Los testers, que tienen un punto de vista diferente al de los desarrolladores y los representantes de las empresas [ISTQB_FL_SYL], aportan una valiosa contribución al intercambio de pensamientos, opiniones y experiencias. La conversación comienza durante la fase de planificación del lanzamiento y continúa cuando la historia está programada
- Confirmación: Los criterios de aceptación, discutidos en la conversación, se utilizan para confirmar que la historia se ha realizado. Estos criterios de aceptación pueden abarcar múltiples historias de usuario. Se deben utilizar pruebas tanto positivas como negativas para cubrir los criterios. Durante la confirmación, varios participantes juegan el papel de testers. Estos pueden incluir desarrolladores, así como especialistas enfocados en el desempeño, seguridad, interoperabilidad y otras características de calidad. Para confirmar que una historia está hecha, los criterios de aceptación definidos deben ser probados y se debe demostrar que están satisfechos.

4.3 Ejemplo y desafíos

OA-4.3	K2	Demuestre el formato con un ejemplo
OA-4.4	K2	Comprender la dificultad al definir los criterios de aceptación al usar el formato.

Se proporciona un ejemplo. Una vez que los estudiantes han comprendido este ejemplo, se enfrentan al hecho de que el ejemplo es defectuoso y por qué: una historia de usuario debe ser lo más específica posible para tener algún valor. Esta especificación es demostrada.

4.4 ¿Qué es un criterio de aceptación?

OA-4.5	K1	Definir un criterio de aceptación
OA-4.6	K2	Comprender el papel de los testers en la definición y elaboración de los criterios de aceptación

El criterio de aceptación se explica en términos del formato (usamos: dado, cuando, luego (sin embargo, no hay reglas, el equipo decide)) y el objetivo (crear límites, consenso sobre qué (y qué no) hacer, y es una base para probar y estimar el esfuerzo y la planificación). Para ser probables, los criterios de aceptación deben abordar los siguientes temas cuando sea pertinente **[WG1]:**

- Comportamiento funcional: El comportamiento externamente observable con las acciones del usuario como entrada operando bajo ciertas configuraciones.
- Características de calidad: Cómo el sistema realiza el comportamiento especificado. Las características también pueden ser referidas como atributos de calidad o requisitos no funcionales. Las características de calidad comunes son el rendimiento, la fiabilidad, la facilidad de uso, 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 operativa.
- Reglas del sistema: 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: Descripción 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 restrinja las opciones para el desarrollador. Los dispositivos con software incorporado deben respetar a menudo las 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 compleja de datos comerciales (por ejemplo, el código postal en una dirección de correo de los Estados Unidos).

El papel de un tester es utilizar todos sus conocimientos y experiencia para conseguir que el equipo sea consciente de que tiene criterios de aceptación con un claro valor comercial.

Los criterios de aceptación son difíciles de describir de manera no ambigua y para ello se necesita la ayuda de un tester. Pensar en los criterios de aceptación mejora las historias de usuario.

4.5 Cuentos de terror

OA-4.7	K2	Comprender los desafíos que implica la definición de los criterios de aceptación
OA-4.8	K2	Entender los cuentos de terror como un medio para encontrar criterios de aceptación

Los cuentos de terror se introducen como un medio para superar el inconveniente de la forma habitual de encontrar criterios de aceptación (al ser una lista no concluyente).

Los cuentos de terror son para abordar las historias de usuario desde un punto de vista de prueba específico, donde la pregunta principal es: "¿qué puede salir mal?".

La definición de un cuento de terror: "el resultado de una historia de usuario que definitivamente no quieres que tus clientes descubran o vean". Un ejemplo: puedo ver los datos de la cuenta bancaria de otra persona, como un cuento de terror de acceso a mi entorno bancario personal.

Los cuentos de terror se muestran basados en la historia del usuario, como se ha presentado anteriormente.

4.6 División de la historia

OA-4.9	K2	Entender la división de la historia
OA-4.10	K2	Comprender cómo dividir las historias
OA-4.11	K1	Recordar cuándo dividir una historia
OA-4.12	K2	Entender cuando una historia está bien dividida.
OP-4.1	OP-3	Analizar y rediseñar la historia dada

Este capítulo comienza con una historia que es demasiado grande para construir / entender / planificar. A continuación, se explican tres formas de dividir una historia (flujo de trabajo, acciones de datos, roles de usuario). Luego se explica cuándo dividir las historias y terminamos con los criterios utilizados para comprobar si una historia es de la calidad adecuada (se explica en detalle el acrónimo INVEST (Independiente, Negociable, Valioso, Estimable, Sencillo y Testeable).

Gráfico 6 ***[AF1]***

Finalmente, los estudiantes se enfrentan a una historia de usuario que requiere que apliquen los conocimientos obtenidos en este módulo. Necesitan dividir la historia, definir nuevas historias con el formato correcto; ser invitados pasivamente a encontrar nuevos criterios de aceptación (usando cuentos de terror), y también a comprobar sus propias historias de usuario más sencillas definidas con el acrónimo INVEST.

Capítulo 5 - Estrategia de prueba

Basándose en los riesgos, el tester (y el equipo) deben decidir elegir una forma efectiva y apropiada de probar, una forma que se ajuste a los riesgos, al proyecto, a la empresa y a sus objetivos, al equipo y al producto. La estrategia debe dejar claro qué vamos a hacer, por qué lo hacemos, cómo lo hacemos, quién lo hace y cuáles han sido los resultados de nuestras acciones. Esto le da al propietario del producto (Interesados /DP) las mejores perspectivas posibles en el software y, con base a estas perspectivas, pueden decidir qué hacer a continuación. Las primeras partes se tratan en los capítulos 2 y 3. Este capítulo contiene las partes finales de los objetos de estrategia de prueba.

Palabras clave

Ágil, Prueba, Estrategia de prueba, Prueba exploratoria, Revisión, carta de prueba, Automatización de prueba, Área de prueba.

OA-5.1	K2	Comprender que una estrategia de prueba es algo más que la fase de "pensar y construir" en el desarrollo. Las pruebas son una actividad constante, una calidad continua.
OA-5.2	K2	Comprender que la información valiosa sobre el producto provendrá de la situación de producción y puede ayudarle a crear una mejor comprensión de los riesgos y puede ayudarle a ajustar su estrategia de prueba.
OA-5.3	K2	Comprender la diferencia entre los diferentes tipos de pruebas; revisión versus intuición y exploración
OA-5.4	K2	Ser capaz de demostrar los niveles de pruebas mapeados en el "Hay conocimientos conocidos" de Rumsfeld
OA-5.5	K2	Comprender la definición de BDD (Behaviour Driven Development), desarrollo dirigido por el comportamiento
OA-5.6	K2	Comprender la definición de SBE (Specification By Example), especificación a través del ejemplo
OA-5.7	K2	Comprender la definición de (A)TDD ((Acceptance) Test Driven Development) ((Aceptación) desarrollo dirigido por pruebas)
OA-5.8	K2	Comprender las relaciones entre BDD/(A)TDD/SBE
OA-5.9	K2	Comprender la sintaxis de la especificación Gherkin
OA-5.10	K2	Comprender la relación entre BDD/(A)TDD/SBE, Gherkin y la automatización de pruebas
OA-5.11	K2	Comprender las sesiones de 3-amigo y el mapeo de ejemplo
OP-5.1	OP-2	Poder utilizar el mapeo de SBE y de ejemplo
OA-5.12	K1	Recordar la definición de pruebas exploratorias según Cem Kaner
OA-5.13	K3	Usar la definición de prueba exploratoria para ejecutar una prueba exploratoria
OA-5.14	K2	Comprender las diferentes fuentes y posibilidades de establecer pruebas exploratorias
OA-5.15	K1	Ser capaz de seleccionar las fuentes apropiadas para ejecutar una prueba exploratoria
OA-5.16	K2	Comprender lo que es una carta de prueba
OA-5.17	K2	Comprender cómo realizar una sesión de prueba exploratoria
OA-5.18	K2	Comprender cómo documentar sus sesiones de pruebas exploratorias
OA-5.19	K2	Comprender la definición de automatización de pruebas
OA-5.20	K1	Recordar revisión versus intuición y exploración
OA-5.21	K2	Comprender el valor de la automatización de las pruebas
OA-5.22	K2	Comprender las limitaciones de la automatización de las pruebas

OA-5.23	K2	Clasificar diferentes tipos de pruebas en la automatización de pruebas
OA-5.24	K3	Aprende a aplicar la pirámide de automatización de pruebas
OA-5.25	K2	Demostrar la diferencia entre la pirámide de automatización de pruebas y el cono de hielo anti-patrón de automatización de pruebas
OA-5.26	K2	Comprender la importancia de estar familiarizado con una amplia gama de herramientas que pueden hacer sus pruebas más eficientes
OA-5.27	K2	Comprender el valor y la necesidad de la automatización en un contexto ágil

5.1 Panorama general

OA-5.1	K2	Comprender que una estrategia de prueba es algo más que la fase de "pensar y construir" en el desarrollo. Las pruebas son una actividad constante, una calidad continua
OA-5.2	K2	Comprenda que la información valiosa sobre el producto provendrá de la situación de producción y puede ayudarle a crear una mejor comprensión de los riesgos y puede ayudarle a ajustar su estrategia de prueba.
OA-5.3	K2	Comprender la diferencia entre los diferentes tipos de pruebas; revisión versus intuición y exploración
OA-5.4	K2	Ser capaz de demostrar los niveles de pruebas mapeados en el "Hay conocimientos conocidos" de Rumsfeld

Primero se explora el dominio de diferentes tipos de pruebas. Hay muchos tipos de pruebas y cada una tiene su propio objetivo y valor. En el pasado, los testers se centraban principalmente en los tipos de pruebas relacionadas con los controles; pruebas que nos dicen algo sobre los hechos que conocemos del sistema (por la documentación). Comprobamos si el sistema hace lo que se supone que debe hacer, ¿funciona?

Desde la introducción de ágil, nuestros clientes se convierten en el centro de todo lo que hacemos. Los equipos se centran en si el sistema/producto hace lo que el cliente espera. Este es un enfoque fundamentalmente diferente que sólo se cubre parcialmente cuando comprobamos si el sistema funciona correctamente. Además, los clientes hoy en día esperan que un sistema sea de buena calidad. Las fallas pueden resultar en que los clientes se trasladen a un competidor que pueda entregar el mismo producto que no falla.

Las pruebas en ágil comienzan en la fase de " piensa" / " diseña" convirtiéndote en un pensador crítico que cuestiona y desafía los requisitos y llega a los cuentos de terror. A esto se le llama normalmente " desplazamiento a la izquierda”.

Además, durante el envío y especialmente durante la producción, realizamos "pruebas" reuniendo datos de vigilancia y registro para obtener información sobre el uso del producto. También podemos utilizar pruebas como las pruebas A/B para obtener información sobre nuestras nuevas características. Normalmente esto se denomina "desplazamiento a la derecha".

Los equipos aprenden y se adaptan constantemente, basándose en los resultados de todas las pruebas. Ajustamos los riesgos y nuestra estrategia todo el tiempo.

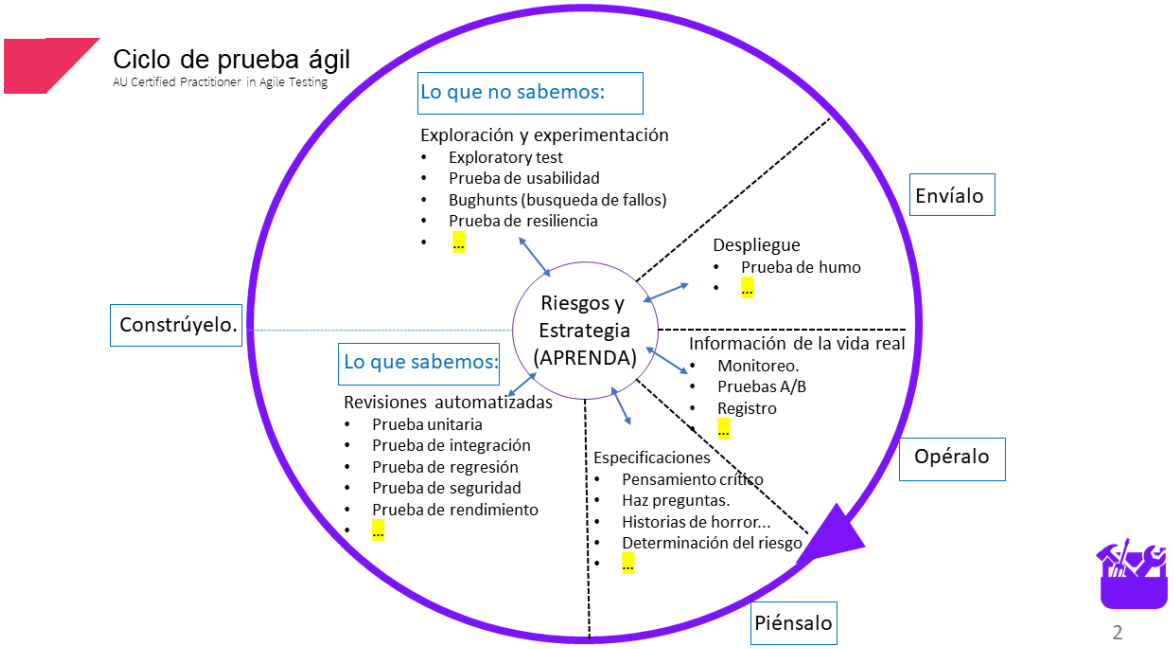


Gráfico 7

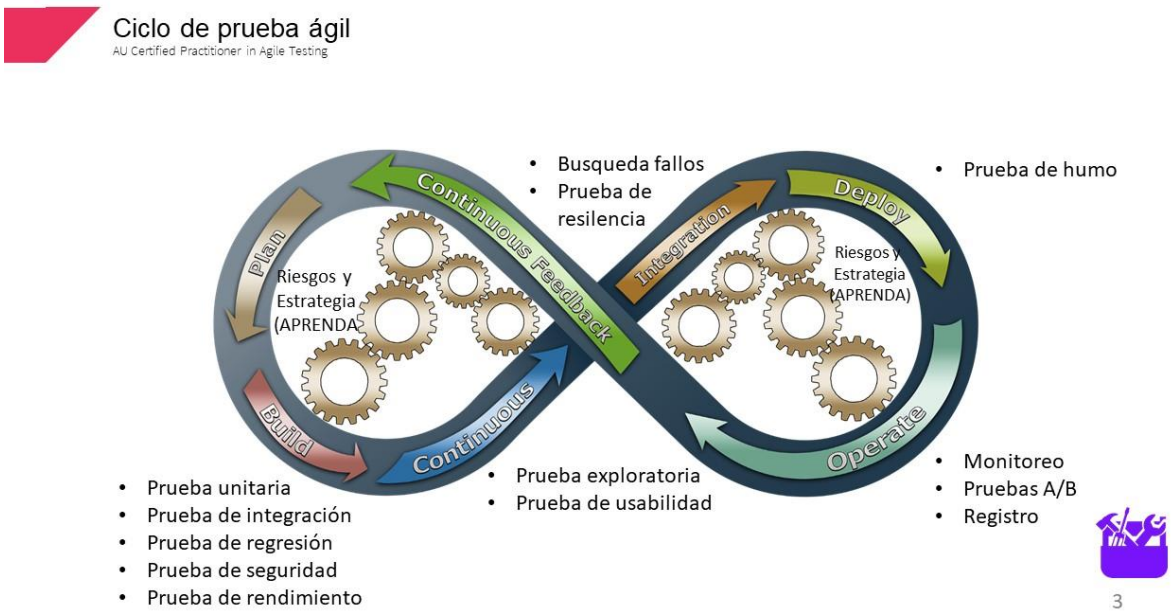


Gráfico 8

Las pruebas se han convertido en una actividad constante que comienza con el " Piensa / Diseña " y llega hasta la producción. El propósito del Ciclo de Pruebas Ágiles es que usted trace sus propios tipos de pruebas. Así que, ¿qué pruebas va a hacer y cuándo exactamente? Esto depende del contexto y los ejemplos anteriores son una interpretación. El propósito del modelo es que usted mismo empiece a pensar qué pruebas debe hacer y cuándo es el momento más conveniente para hacerlas.

5.2 Desarrollo impulsado por el comportamiento

OA-5.5	K2	Comprender la definición de BDD (Behaviour Driven Development)
OA-5.6	K2	Comprender la definición de SBE (Specification By Example)
OA-5.7	K2	Comprender la definición de (A)TDD ((Acceptance) Test Driven Development)
OA-5.8	K2	Comprender las relaciones entre BDD/(A)TDD/SBE
OA-5.9	K2	Comprender la sintaxis de la especificación Gherkin
OA-5.10	K2	Comprender la relación entre BDD/(A)TDD/SBE, Gherkin y la automatización de pruebas
OA-5.11	K2	Comprender las sesiones de 3-amigos y el mapeo de ejemplo
OP-5.1	OP-2	Poder utilizar el mapeo de SBE y de ejemplo

En la ingeniería de software, el “Desarrollo dirigido por el comportamiento” (BDD) es un proceso ágil de desarrollo de software que fomenta la colaboración entre los desarrolladores, control de calidad y los participantes no técnicos o directos en un proyecto de software **[DN1]**. Alienta a los equipos a utilizar la conversación y los ejemplos concretos para formalizar un entendimiento compartido de cómo debe comportarse la aplicación. Este concepto surgió a partir del desarrollo basado en pruebas (TDD) **[KB1]**. El desarrollo basado en el comportamiento combina las técnicas y principios generales del TDD con ideas del diseño basado en el dominio **[EE1]** y el análisis y diseño orientado a objetos para proporcionar a los equipos de desarrollo y gestión de software, herramientas compartidas y un proceso compartido para colaborar en el desarrollo de software.

TDD es una metodología de desarrollo de software que esencialmente establece que, para cada unidad de software, un desarrollador de software debe:

- definir primero un conjunto de pruebas para la unidad.
- hacer que las pruebas fallen.
- luego implementar la unidad.
- finalmente verificar que la implementación de la unidad hace que las pruebas tengan éxito.

Esta definición es bastante inespecífica en el sentido de que permite pruebas en términos de requisitos de software de alto nivel, detalles técnicos de bajo nivel o cualquier otra cosa entre ellas. Por lo tanto, una forma de ver la BDD es que es un desarrollo continuo de TDD lo que hace que las opciones sean más específicas que la TDD.

BDD especifica que las pruebas de cualquier unidad de software deben ser especificadas en términos del comportamiento deseado de la unidad. Tomando prestado del desarrollo ágil de software el "comportamiento deseado" en este caso consiste en los requisitos establecidos por la empresa, es decir, el comportamiento deseado que tiene valor comercial para cualquier entidad que haya encargado la unidad de software en construcción.
Siguiendo esta elección fundamental, una segunda elección hecha por BDD se refiere a cómo debe especificarse el comportamiento deseado. En esta área BDD elige utilizar un formato semiformal para la especificación del comportamiento que se toma prestado de las especificaciones de la historia del usuario del campo del análisis y diseño orientado al objeto.
BDD especifica que los analistas y desarrolladores deben colaborar en esta área y deben especificar el comportamiento en términos de historias de usuario. Cada historia de usuario debe, de alguna manera, seguir la siguiente estructura:

- Título
  Un título explícito.
- Narrativa
  Una breve sección introductoria con la siguiente estructura:

Como: la persona o el papel que se beneficiará de la característica;
Quiero: las características;
de modo que: el beneficio o valor del rasgo.

- Criterios de aceptación
  Una descripción de cada escenario específico de la narración con la siguiente estructura:
  Dado: el contexto inicial al principio del escenario, en una o más cláusulas;
  Cuando: el evento que desencadena el escenario;
  Después: el resultado esperado, en una o más cláusulas.

BDD no tiene ningún requisito formal sobre la forma exacta en que deben escribirse estas historias de usuarios, pero insiste en que cada equipo que utilice BDD elabore un formato sencillo y estandarizado para escribir las historias de usuario que incluya los elementos enumerados anteriormente.

Especificación mediante ejemplo (SBE) **[GA1]** es un enfoque de colaboración para definir los requisitos y las pruebas funcionales orientadas a la empresa para los productos de software, basado en la captura e ilustración de los requisitos mediante ejemplos realistas en lugar de declaraciones abstractas. Se aplica en el contexto de los métodos ágiles de desarrollo de software, en particular BDD. Este enfoque es particularmente exitoso para la gestión de requisitos y pruebas funcionales en proyectos a gran escala de dominio y complejidad organizativa significativa.

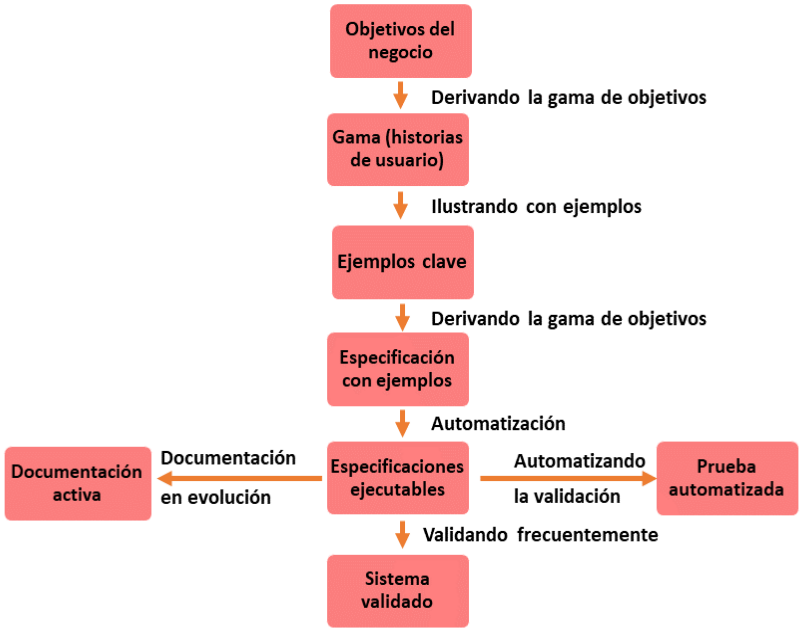
Un aspecto clave de SBE es crear una única fuente de verdad sobre los cambios requeridos desde todas las perspectivas. Cuando los analistas trabajan en sus propios documentos, los desarrolladores de software mantienen su propia documentación y los testers mantienen un conjunto separado de pruebas funcionales, la eficacia de la entrega del software se reduce significativamente por la necesidad de coordinar y sincronizar constantemente esas diferentes versiones de la verdad. Con SBE, los diferentes papeles participan en la creación de una única fuente de verdad que capta la comprensión de todos. Los ejemplos se utilizan para proporcionar claridad y precisión, de modo que la misma información pueda utilizarse tanto como una especificación como una prueba funcional orientada a los negocios. Toda información adicional descubierta durante el desarrollo o la entrega, como la aclaración de las lagunas funcionales, los requisitos que faltan o están incompletos o las pruebas adicionales, se añade a esta única fuente de verdad. Como sólo hay una fuente de verdad sobre la funcionalidad, no es necesario coordinar, traducir e interpretar los conocimientos dentro del ciclo de entrega.

Cuando se aplica a los cambios requeridos, un conjunto de ejemplos refinados es efectivamente una especificación y una prueba orientada a los intereses para la aceptación de la funcionalidad del software. Una vez implementado el cambio, la especificación con ejemplos se convierte en un documento que explica la funcionalidad existente. Como la validación de esos documentos es automática, cuando se validan con frecuencia, esos documentos son una fuente fiable de información sobre la funcionalidad del software subyacente.

Los Tres Amigos **[AA2]**, también conocido como "Taller de Especificaciones", es una reunión en la que el propietario del Producto discute el requerimiento en forma de SBE, por Ejemplo, con diferentes interesados, como los miembros del equipo. El objetivo principal de esta discusión es iniciar la conversación e identificar cualquier especificación que falte. El debate también ofrece una plataforma para que el equipo y el propietario del producto converjan y escuchen la perspectiva de cada uno para enriquecer el requisito y también para asegurarse de que están elaborando el producto correcto.

Los tres Amigos son

- Negocio - El papel del usuario de negocios es definir el problema solamente (y no aventurarse a sugerir ninguna solución).
- Desarrollo - El papel de los desarrolladores consiste en sugerir formas de solucionar el problema.
- Pruebas - El papel de los testers es cuestionar la solución, plantear tantas posibilidades como sea posible para la tormenta de cerebros a través de los escenarios Y si... y ayudar a hacer la solución más precisa para solucionar el problema.



Gráfica 9

El refinamiento de la especificación puede hacerse con el ejemplo del mapeo. **[MW1]** El Ejemplo de mapeo es una técnica que puede dirigir la conversación y derivar criterios de aceptación en un plazo de 30 minutos. El proceso consiste en dividir cada historia en Reglas y Ejemplos y documentarla en forma de Especificaciones por medio de ejemplos.

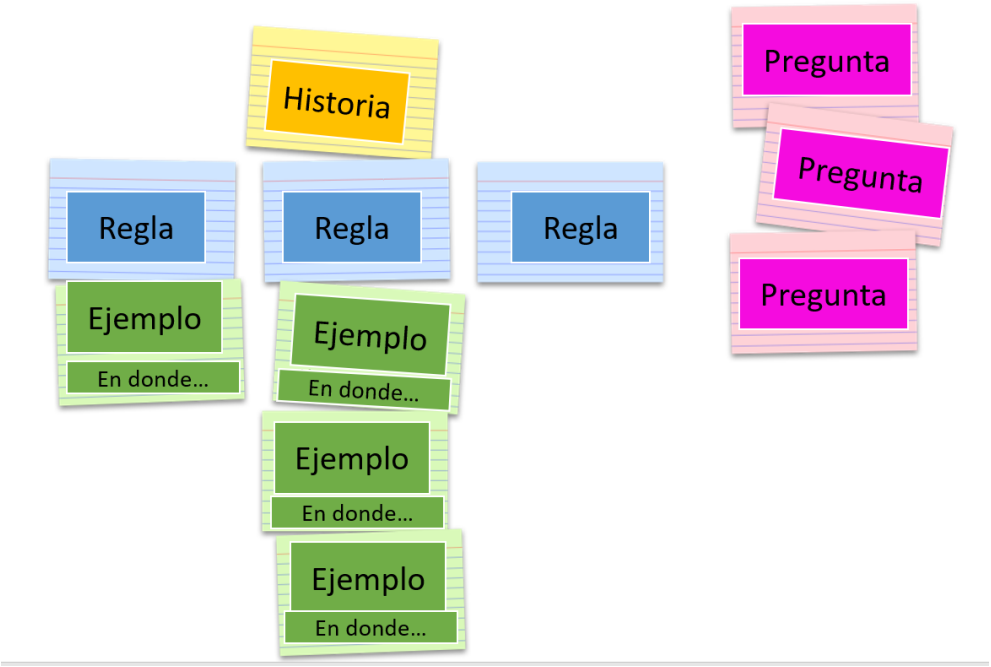


Gráfico 10

Para hacer posible la automatización (que por sí misma no es el objetivo de SBE) es necesario escribir las especificaciones de una manera ejecutable. Gherkin **[GH1]** es el lenguaje que utiliza Cucumber para definir los casos de prueba. Está diseñado para no ser técnico y legible por el ser humano, y describe colectivamente casos de uso relacionados con un sistema de software. El propósito de la sintaxis de Gherkin es promover las prácticas de BDD a través de todo un equipo de desarrollo, incluyendo analistas y gerentes. Trata de hacer cumplir requisitos firmes e inequívocos, comenzando en las fases iniciales de la definición de los requisitos por la dirección de la empresa y en otras etapas del ciclo de vida del desarrollo.

Además de proporcionar un guion para la prueba automatizada, la sintaxis del lenguaje natural de Gherkin está diseñada para proporcionar una documentación sencilla del código que se está probando.

Las pruebas de Cucumber se dividen en características individuales. Estas Características se subdividen en escenarios, que son secuencias de Pasos.

Una Característica es un Caso de Uso que describe una función específica del software que se está probando. Hay tres partes en una Característica.

- The Feature (La característica): palabra clave
- El nombre del Feature (la característica), en la misma línea que la palabra clave
- Una descripción opcional en las siguientes líneas

Ejemplo Definición del “Feature”:

```
Feature: Withdraw Money from ATM
```

```
A user with an account at a bank would like to withdraw money from an ATM.

Provided he has a valid account and debit or credit card, he should be
allowed to make the transaction. The ATM will tend the requested amount of
money, return his card, and subtract amount of the withdrawal from the
user's account.

    Scenario: Scenario 1
        Given preconditions
        When actions
        Then results

    Scenario: Scenario 2
        ...
```

Cada característica está hecha de una colección de "Scenarios" (escenarios). Un único escenario es un flujo de eventos a través de la Característica que se describe y mapea 1:1 con un caso de prueba ejecutable para el sistema. Siguiendo con el ejemplo de la característica de retiro de los cajeros automáticos, un escenario podría describir cómo un usuario solicita dinero y qué sucede con su cuenta.

```
Scenario: Eric wants to withdraw money from his bank account at an ATM
    Given Eric has a valid Credit or Debit card
    And his account balance is $100
    When he inserts his card
    And withdraws $45
    Then the ATM should return $45
    And his account balance is $55
```

El punto crucial de un Escenario se define por una secuencia de Pasos que describen las condiciones previas y el flujo de eventos que tendrán lugar. La primera palabra de un paso es una palabra clave, normalmente una de;

- "Given" (Dado)- Describe las condiciones previas y el estado inicial antes del comienzo de una prueba y permite cualquier configuración previa a la prueba que pueda ocurrir
- "When" (Cuando) - Describe las acciones realizadas por un usuario durante una prueba
- "Then" (Después) - Describe el resultado de las acciones tomadas en la cláusula Cuando

Ocasionalmente, la combinación de Dado-Cuando-Después utiliza otras palabras clave para definir las conjunciones;

- - "And" (Y) - Lógico y
- - "But" (Pero) - Lógicamente lo mismo que "And", pero usado en la forma negativa

```
Scenario: A user attempts to withdraw more money than they have in their
account
    Given John has a valid Credit or Debit card
    And his account balance is $20
    When he inserts his card
    And withdraws $40
    Then the ATM displays an error
    And returns his card
```

```
    But his balance remains $20
```

5.3 Pruebas exploratorias

OA-5.12	K1	Recordar la definición de pruebas exploratorias según Cem Kaner
OA-5.13	K3	Usar la definición de prueba exploratoria para ejecutar una prueba exploratoria
OA-5.14	K2	Comprender las diferentes fuentes y posibilidades de establecer pruebas exploratorias
OA-5.15	K1	Ser capaz de seleccionar las fuentes apropiadas para ejecutar una prueba exploratoria
OA-5.16	K2	Comprender lo que es una carta de prueba
OA-5.17	K2	Comprender cómo establecer una sesión de prueba exploratoria
OA-5.18	K2	Comprender cómo documentar sus sesiones de pruebas exploratorias

Con las pruebas exploratorias, se introduce una forma inicial de pruebas de cara al cliente. El producto se explora con base a un cliente que se enfrenta a riesgos y, al hacerlo, se tocan áreas de la aplicación que son desconocidas ya que no pudimos documentarlo porque no lo conocíamos.

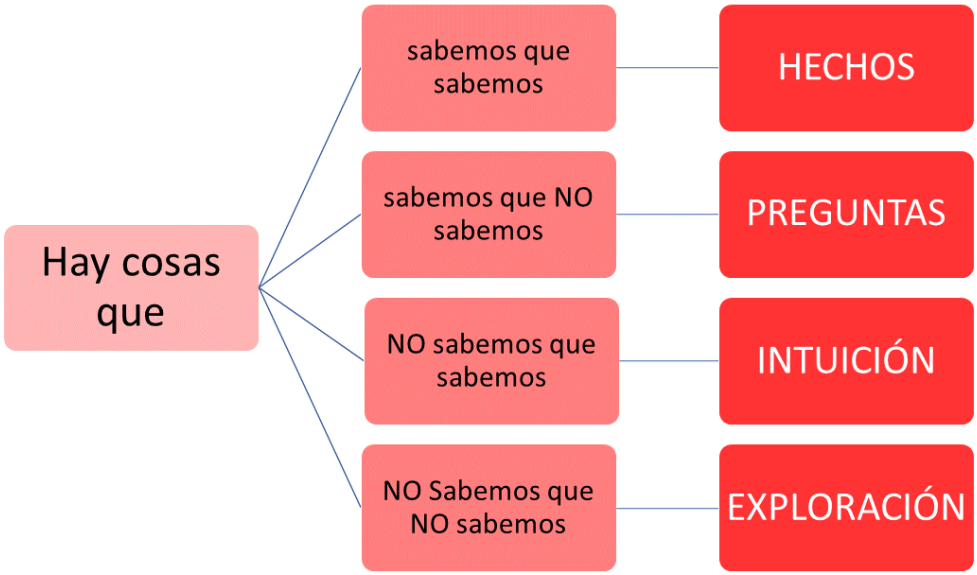


Figure 11

Las pruebas exploratorias se explican según la definición de Cem Kaner:

"La prueba exploratoria de software es un estilo de prueba de software que hace hincapié en la libertad personal y la responsabilidad del tester individual de optimizar continuamente el valor de su trabajo tratando el aprendizaje relacionado con la prueba, el diseño de la prueba, la ejecución de la prueba y la interpretación de los resultados de la prueba como actividades que se apoyan mutuamente y que se desarrollan en paralelo a lo largo del proyecto" ***[CK1]***

Explicamos las ventajas de las pruebas exploratorias en términos de:

- Obtención de conocimientos en las incógnitas
- Aprende sobre el producto muy rápido
- Conoce las características no especificadas y el uso del sistema
- Prueba mucho más rápida y efectivamente
- Trae de vuelta lo desconocido al dominio de lo conocido

Después de la definición se introduce el primer enfoque hacia las pruebas exploratorias. Cuales elementos juegan un papel en las pruebas exploratorias y como un tester puede empezar a hacer valiosas pruebas exploratorias.

Los elementos típicos que pueden desempeñar un papel en las pruebas exploratorias son:

- Esquema del proyecto
- Esquema del producto
- Riesgos
- Atributos de calidad
- Especificaciones
- Impacto técnico
- Intuición/Experiencia
- Mnemotécnica
- Heurística
- Listas de verificación
- Patrones históricos de fracaso

La carta de prueba se introduce como una forma de estructurar y guiar las pruebas exploratorias y la hoja de sesión se añade a ella para anotar los resultados de nuestras pruebas. La carta de prueba y la hoja de sesión consisten en:

- Carta de prueba
    - Misión de nuestra prueba (lo que estamos tratando de lograr).
    - Propósito de nuestra prueba (cómo vamos a lograrlo, qué estamos buscando, qué esperamos encontrar, dónde esperamos encontrarlo).
- Reporte de sesión
    - Periodo
    - Notas
        - Lo que estamos probando
        - Los resultados de la prueba
        - Las preguntas que se plantean
        - Problemas encontrados
        - Los datos relevantes de las pruebas que hemos usado/necesitado
    - Información para el informe BRIEF
        - **B**ehaviour (Comportamiento): como se comportó el software durante nuestra sesión
        - **R**esults (resultados): lo que logramos
        - **I**mpediments (Impedimentos): lo que estaba en nuestro camino
        - **E**xpectations (Expectativas): lo que aún queda por hacer
        - **F**eelings (Sentimientos): ¿cómo me siento sobre el producto?

5.4 Automatización de pruebas y herramientas

OA-5.19	K2	Comprender la definición de automatización de pruebas
OA-5.20	K1	Comprobación de la memoria versus intuición y exploración
OA-5.21	K2	Comprender el valor de la automatización de las pruebas
OA-5.22	K2	Comprender las limitaciones de la automatización de las pruebas
OA-5.23	K2	Clasificar diferentes tipos de pruebas en la automatización de pruebas
OA-5.24	K3	Aprender a aplicar la pirámide de automatización de pruebas
OA-5.25	K2	Demostrar la diferencia entre la pirámide de automatización de pruebas y el cono de hielo anti-patrón de automatización de pruebas
OA-5.26	K2	Comprender la importancia de estar familiarizado con una amplia gama de herramientas que pueden hacer sus pruebas más eficientes
OA-5.27	K2	Comprender el valor y la necesidad de la automatización en un contexto ágil

La definición de la automatización de las pruebas: "una forma de repetir automáticamente la ejecución de los casos de prueba, incluyendo las comprobaciones de los resultados predefinidos (validaciones automáticas)".

Esto también proporciona las limitaciones de la automatización de las pruebas:

- Necesitamos resultados predefinidos que no cambien
- Necesitamos crear guiones de prueba por adelantado, así tiene que estar claro lo que el sistema hará
- Sólo cubrimos los hechos (verificaciones), no la intuición y la exploración
- ¿Hay suficiente valor en repetir las mismas pruebas una y otra vez?
- El retorno de la inversión en la automatización de las pruebas debe tenerse en cuenta

Se muestran los diferentes tipos de niveles en los que la automatización de las pruebas puede jugar un papel utilizando la pirámide de automatización de pruebas introducida por Mike Cohn **[MC1]** que introduce 3 capas de automatización de pruebas:

- Nivel UI (interfaz)
- API o nivel de servicio
- Nivel de la unidad

El valor que las pruebas automatizadas entregan es:

- Una prueba unitaria automatizada entrega la mayor parte del valor ya que el ciclo de retroalimentación es muy corto y muchas verificaciones pueden cubrirse fácilmente.
- Las pruebas de API o de nivel de servicio (las pruebas por contrato son un término común hoy en día) son valiosas, así como una API tiene un contenido predefinido muy claro y eso la hace adecuada para la comprobación automatizada.
- El último nivel que debería contener menos pruebas es el nivel de interfaz de usuario. Este nivel está cambiando mucho y las pruebas automatizadas son deficientes y difíciles de mantener y las herramientas son caras de comprar o de configurar y mantener.

Desafortunadamente, como el nivel de UI es el más visible, muchas organizaciones eligen principalmente crear pruebas automatizadas al nivel de UI. Esto es lo que se llama el "cono de hielo" anti-patrón de la automatización de pruebas. Tener esta información le da al tester la oportunidad de explicar y mostrar el beneficio de una buena estrategia de automatización de pruebas **[MC1]**.

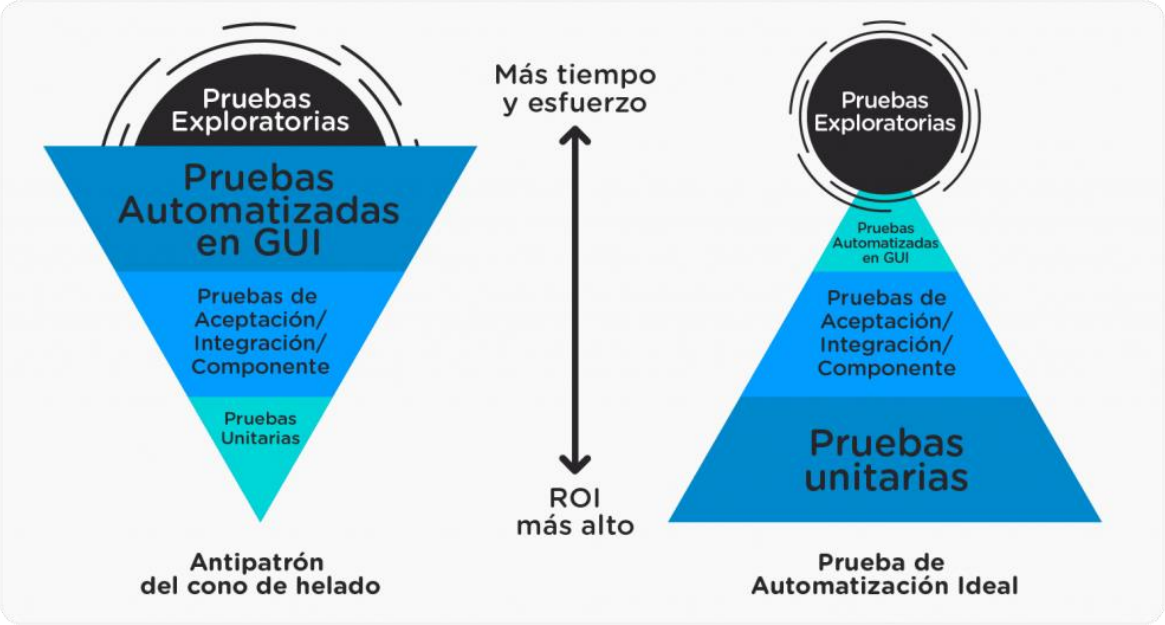


Gráfico 12

La gente conoce las herramientas de prueba, sin embargo, pensamos principalmente en ellas como herramientas de automatización de pruebas como Selenium, SoapUI, Citrus, Cypress...go

Hay muchas más herramientas que pueden ser muy útiles a veces, para ayudar a los testers a:

- Documentar nuestras pruebas, como grabadoras de pantalla o incluso Word o un wiki o Confluence
- Encontrar defectos potenciales más fácilmente, como Firebug o Bug Magnet
- Ejecutar algunos primeros controles de seguridad básicos, como ZAP
- Identificar los primeros problemas potenciales de rendimiento, como Yslow
- Comprobar los posibles problemas de uso, como el WCAAG A-checker

Capítulo 6 - Informe de la prueba

Es necesario que se reporte lo que se hace en relación con la calidad y las pruebas. ¿Qué se hizo y con qué resultados? Los reportes deben ser sencillos, ya que valoramos más el funcionamiento del software que la documentación exhaustiva. Por lo general, la presentación de informes también es necesaria para los fines de la gobernanza, el riesgo y el cumplimiento (GRC). La presentación de reportes depende en gran medida del contexto de la organización, el proyecto específico y el producto. Siempre se aplica una regla: la presentación de reportes debe satisfacer una necesidad. La cantidad y la forma de presentación de reportes depende de esta necesidad.

Existen tres formas diferentes de informar: el resumen de estado de una página, el relato de historias y la gestión de problemas.

Palabras clave

Ágil, Probando, Reportando

OA-6.1	K2	Comprender el valor de reportar sobre las pruebas y la calidad en ágil
OA-6.2	K3	Poner en práctica el informe sobre los aspectos de ensayo y calidad
OP-6.1	OP-2	Crear reportes reales y valiosos sobre el tema de la prueba
OA-6.3	K1	Recordar por qué la gente necesita que le cuenten historias. De niños aprendemos de las historias, de adultos hemos olvidado esta forma tan efectiva de aprender y reportar
OA-6.4	K2	Entender lo que es realmente contar una historia y ser capaz de explicarla a los demás
OA-6.5	K2	Demostrar los ingredientes mínimos de una buena historia
OA-6.6	K3	Aplicar la narración de historias a las pruebas y ser capaz de contar una historia de pruebas
OP-6.2	OP-1	Aprende a contar una historia guiada por los pasos de los formadores
OA-6.7	K1	Ser capaz de recordar la definición de un problema o defecto
OA-6.8	K2	Comprender cuándo informar de un incidente y cuándo no
OA-6.9	K2	Comprender cómo informar de un incidente para que sea de valor
OP-6.3	OP-2	Crear un informe sobre el software probado según los principios aprendidos

6.1 El resumen de una página sobre el estado de la prueba

OA-6.1	K2	Comprender el valor de informar sobre las pruebas y la calidad en ágil
OA-6.2	K3	Poner en práctica el reporte A3 sobre los aspectos de prueba y calidad
OP-6.1	OP-2	Crear informes reales y valiosos sobre el tema de la prueba

Ser ágil significa que los equipos se centran menos en la documentación. Sin embargo, esto no significa que no haya necesidad de ninguna documentación. Todavía tendremos que ser capaces de reportar nuestros progresos de forma regular e instantánea. Se identifican los elementos que son

importantes y se dan algunos consejos visuales de cómo utilizarlos para ofrecer el valor necesario a los clientes.

Un ejercicio para crear realmente un reporte de prueba A3 sobre las pruebas que se acaban de realizar termina este capítulo.

La información típica sobre la que se debe reportar:

- Lo que se ha probado y lo que no se ha probado (todavía) y por qué no.
- Cuáles fueron los resultados de las pruebas, la información que contiene la prueba.
- Quien realizó las pruebas y el lapso de tiempoen el que se llevaron acabo.
- ¿Qué necesita más pruebas y/o tiene mayor riesgo?
- De qué manera se incorporan las regulaciones (si es aplicable)

6.2 La historia de la prueba / La narración de la historia

OA-6.3	K1	Recordar por qué la gente necesita que le cuenten historias. De niños aprendemos de las historias, de adultos hemos olvidado esta forma tan efectiva de aprender y reportar
OA-6.4	K2	Entender lo que es realmente contar una historia y ser capaz de explicarla a los demás
OA-6.5	K2	Demostrar los ingredientes mínimos de una buena historia
OA-6.6	K3	Aplicar la narración de historias a las pruebas y ser capaz de contar una historia de pruebas
OP-6.2	OP-1	Aprende a contar una historia guiada por los pasos de los formadores

En la era pre-ágil, los testers creaban "valor visible" creando reportes de prueba sobre los planes de prueba creados, el número de pruebas creadas y ejecutadas, el número de defectos encontrados. Se creó mucha documentación y eso convenció a la mayoría de los directivos de que un tester añadía un valor real.

En la era ágil, creamos menos documentación. Nos centramos mucho más en dar información valiosa sobre la calidad de la aplicación, en evitar que se produzcan defectos y en la colaboración del equipo cuando se encuentran defectos para resolverlos al instante en lugar de sólo informarlos.

Se añade algo de neurociencia a la narración de historias **[ST1]** para que entienda por qué algunas historias son más interesantes de escuchar que otras:

- La dopamina: hace que la gente esté más motivada y activa la memoria.
- Oxitocina: hace a la gente más generosa y desencadena la necesidad de conectarse.
- Endorfinas: hace a la gente más creativa y relajada.

¿Cómo se asegura de que su audiencia las crea, para que su historia sea interesante de escuchar y su mensaje sea recordado?

- Dopamina: cuente una historia emocionante con suspenso.
- Oxitocina: cuente una historia que evoque empatía (o lástima).
- Endorfinas: cuente una historia que haga reír a la gente.

Esto hace mucho más difícil mostrar el valor real de las pruebas a las partes interesadas fuera del equipo. El desarrollo muestra un producto construido en una demostración. ¿Cómo se puede mostrar el valor añadido de las pruebas y la calidad?

Esto se puede hacer contando la historia de las pruebas. Qué pruebas se han hecho, por qué y con qué resultados. Los puntos clave de la historia de las pruebas son el cliente y el valor que le aportan las pruebas. De esta manera, los testers también pueden contar sobre el valor "no visible" que se crea, como ayudar a crear buenas historias de usuarios, ayudar a los desarrolladores a hacer pruebas unitarias, los defectos encontrados y resueltos directamente junto con el desarrollador/DP.

6.3 Defectos y gestión de los defectos

OA-6.7	K1	Ser capaz de recordar la definición de un defecto/bug
OA-6.8	K2	Comprender cuándo reportar un asunto y cuándo no
OA-6.9	K2	Comprender cómo reportar un asunto para que sea de valor
OP-6.3	OP-2	Crear un informe sobre el software probado según los principios aprendidos

El objetivo principal de las pruebas no es encontrar tantos defectos como sea posible. El objetivo principal de las pruebas es dar una idea de la calidad del producto y ayudar a mejorarlo. Al documentar los defectos, el producto no mejorará. Un tester necesita probar tanto como sea posible para obtener información valiosa. Tan pronto como se encuentra un defecto, el tester debe hablar con el desarrollador y/o producto owner sobre el riesgo que supone el defecto encontrado y, como equipo, decidir juntos qué hacer con el defecto. Si tiene que ser resuelto, entonces el equipo lo resuelve inmediatamente y no lo documenta más que una mención en el registro de la prueba. Si no puede ser resuelto inmediatamente, el equipo puede decidir documentar el defecto para una investigación posterior y pedir al DP que añada un elemento de corrección de errores al registro del producto.

En ese caso, proporcione la siguiente información:

- pasos para reproducir
- resultado esperado y resultado real
- datos de prueba utilizados
- entorno de prueba utilizado
- impresiones de pantalla, video, archivos de registro
- prioridad (progreso de la iteración)
- severidad (cuánto daño sería si estuviéramos en vivo)
- lo que ustedes como equipo encuentran útil (y estrictamente necesario)

Utilizando esta forma de manejar los asuntos se introduce una forma de trabajo que se ajusta a ágil. Prevenir problemas sobre encontrar problemas y resolver problemas sobre documentar problemas. Un tester en un contexto ágil podría utilizar la retrospectiva como medio para trabajar hacia esta forma de pensar en el equipo.

Referencias

Referencias generales

Algunos de los elementos de la estrategia de prueba están inspirados en: The heuristic test strategy model by James Bach https://www.satisfice.com/download/heuristic-test-strategy-model

Parte del contenido está inspirado en: Rapid Software Testing by James Back and Michael Bolton https://rapid-software-testing.com/

[SH1] El software se pondrá a disposición de los socios y formadores. Los detalles estarán en el manual del formador. El hardware puede ser ordenado buscando por 'Ventilador de Holograma LED 3D'. Los ventiladores están ampliamente disponibles, sin embargo, son opcionales de usar.

Referencias específicas

Referencia	Fuente
[BT1]	A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives - By L. Anderson, P. W. Airasian, and D. R. Krathwohl (Allyn & Bacon 2001)
[BT2]	https://www.apu.edu/live_data/files/333/blooms_taxonomy_action_verbs.pdf
[JL1]	Agile Testing, A practical guide for testers and agile teams https://agiletester.ca/ More Agile Testing, Learning Journeys for the Whole Team https://agiletester.ca/
[AG1]	https://www.agilealliance.org/agile101/
[AM1]	http://agilemanifesto.org/
[SC1]	https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf
[RF1]	https://en.wikipedia.org/wiki/There_are_known_knowns **DoD News Briefing - Secretary Rumsfeld and Gen. Myers** **Presenter: Secretary of Defense Donald H. Rumsfeld** **February 12, 2002 11:30 AM ED** https://archive.defense.gov/Transcripts/Transcript.aspx?TranscriptID=2636
[GA1]	https://www.growingagile.co.za/
[BO1]	http://www.developsense.com/resources/Oracles.pdf
[JW1]	Jerry Weinberg http://geraldmweinberg.com/
[TE1]	https://www.testingexcellence.com/test-oracles-test-heuristics/
[TS1]	https://www.ministryoftesting.com/testsphere
[EH1]	http://testobsessed.com/2006/12/the-nightmare-headline-game-planning-for-the-unexpected/
[CK1]	http://kaner.com/?p=46
[DN1]	Dan North https://dannorth.net/introducing-bdd/
[KB1]	Beck, Kent (2002-11-08). *Test-Driven Development by Example*. Vaseem: Addison Wesley. ISBN 978-0-321-14653-3.
[EE1]	Evans, Eric (2004). *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley. ISBN 978-032-112521-7.
[GA1]	Adzic, Gojko (2009). *Bridging the Communication Gap: Specification by Example and Agile Acceptance Testing*. Neuri. ISBN 0-9556836-1-0.
[MW1]	Matt Wyne https://www.youtube.com/watch?v=VwvrGfWmG_U
[AA2]	2009: George Dinwiddie https://www.agilealliance.org/glossary/three-amigos/
[GH1]	https://cucumber.io/docs/guides/overview/#what-is-gherkin
[PD1]	The new consultation: developing doctor–patient communication. David Pendleton, Theo Schofield, Peter Tate, Peter Havelock. Oxford University Press, 2003. ISBN 0-19-263288

[HB1]	https://projectaccess.uoregon.edu/teachers/socialemotional/socialskills/SW%20E%20Hamburger%20Method%20of%20Construct%20Criticism%20DONE.pdf
[MC1]	https://www.mountaingoatsoftware.com/blog/the-forgotten-layer-of-the-test-automation-pyramid
[ST1]	https://en.wikipedia.org/wiki/Storytelling
[US1]	https://www.scrum.org/resources/blog/10-tips-product-owners-business-value
[AA1]	https://www.agilealliance.org/glossary/user-story-template/
[KM1]	Thinking, Fast and Slow, D. Kahneman, ISBN: 9780141033570
[JF1]	Ron Jeffries, Ann Anderson, and Chet Hendrickson, “Extreme Programming Installed,” Addison-Wesley Professional, 2000.
[WG1]	Karl Wiegers and Joy Beatty, “Software Requirements, 3e,” Microsoft Press, 2013.
[AF1]	https://agileforall.com/resources/how-to-split-a-user-story/