

AGILE TESTING CONDENSED

A Brief Introduction by
Janet Gregory & Lisa Crispin

Pruebas ágiles condensadas:

Una breve introducción

Janet Gregory y Lisa Crispín

Este libro está a la venta en <http://leanpub.com/agiletesting-condensed>

Esta versión fue publicada el 2019-09-24

ISBN 978-1-9992205-0-1



Este es un [Leanpub](#) libro. Leanpub empodera a autores y editores con el proceso Lean Publishing. [Publicación ajustada](#) es el acto de publicar un libro electrónico en progreso utilizando herramientas livianas y muchas iteraciones para obtener comentarios de los lectores, girar hasta que tenga el libro correcto y generar tracción una vez que lo tenga.

© 2019 Janet Gregory y Lisa Crispin

Contenido

SECCIÓN 1: Cimentaciones	1
Capítulo 1: ¿Qué queremos decir con pruebas ágiles?	2
Modelos de pruebas continuas.	3
Diez principios para pruebas ágiles.	4
Manifiesto de prueba.	5
Definición de pruebas ágiles.	6
Capítulo 2: Enfoque de todo el equipo y pruebas ágiles de la mente colocar	7
Centrarse en la calidad.	8
Cómo los equipos abordan los defectos.	9
Multiples perspectivas	10
Capítulo 3: Planificación de pruebas en contextos ágiles	12
El equipo	12
El producto	13
Planificación a través de niveles de detalle.	14
Planificación de pruebas de regresión.	17
SECCIÓN 2: Enfoques de prueba	18
Capítulo 4: Orientar el desarrollo con ejemplos	19
Métodos basados en ejemplos.	19
Por qué los ejemplos ayudan.	21
Esta es tu base.	22

CONTENIDO

Capítulo 5: Habilitar la colaboración	24
Colaborar con los clientes.	24
Mapeo de impacto.	25
Hacer preguntas	26
Mapeo de ejemplo.	27
Genere confianza utilizando la visibilidad.	28
Capítulo 6: Explorar continuamente	30
Personas, trabajos y roles.	31
Flujos de trabajo y recorridos.	32
Riesgos y valor para el cliente.	33
Explorar en parejas o grupos.	34
Cartas.	34
Ejecutar, aprender, dirigir.	36
Técnicas adicionales.	36
Aproveche las herramientas para una exploración eficaz.	37
Capítulo 7: Prueba de atributos de calidad	38
Definición de atributos de calidad.	38
Mitigar los riesgos colaborando tempranamente.	39
Planificación de pruebas previas al lanzamiento.	41
Planificación para el aprendizaje posterior.	41
Cumplimiento normativo	42
Capítulo 8: Pruebas en DevOps	43
Entrega e implementación continuas.	44
Pruebas en producción.	46
Seguimiento y observabilidad..	47
La nueva tecnología nos trae nuevas capacidades.	48
SECCIÓN 3: Modelos útiles	49
Capítulo 9: Los cuadrantes de pruebas ágiles	50
¿Qué pruebas y en qué orden?	52
Usando los cuadrantes.	53
Definiendo “Listo”	55

CONTENIDO

Encuentre los modelos que se ajusten a su contexto.	56
Capítulo 10: Visualización de una estrategia de automatización de pruebas	57
Utilizando modelos visuales.	57
La clásica pirámide de automatización de pruebas.	58
Pruebas automatizadas como documentación viva.	60
Ampliando el modelo.	61
Responsabilidad compartida	62
SECCIÓN 4: Pruebas ágiles hoy	63
Capítulo 11: El nuevo rol de un probador	64
Los evaluadores son pegamento de calidad para un equipo.	64
"El recorrido profesional de un tester ágil".	66
El fascinante camino de evolucionar como testers.	68
Se todo lo que puedas ser	70
Comience con una conversación.	71
El mundo no necesita más damas.	72
Los pensamientos de Lisa y Janet.	74
Capítulo 12: Ingredientes para el éxito	76
Factores de éxito	76
Prácticas de fomento de la confianza.	81
Caminos hacia el éxito.	84
Glosario	87
Recursos para un mayor aprendizaje	89
General	89
Obtener comprensión compartida: colaboración.	89
Prueba exploratoria	90
DevOps, Monitoreo, Observabilidad.	90
Automatización de pruebas.	91
Sobre los autores	93

Expresiones de gratitud

Nuestros libros implican siempre un esfuerzo comunitario. Muchas gracias a nuestros colegas que contribuyeron con sus visiones sobre cómo está evolucionando el rol del evaluador en el Capítulo 11. Vamos a escribir un libro corto aquí, por lo que no podemos agradecer individualmente a todas las personas de quienes seguimos aprendiendo nuevas ideas y conceptos para aplicar. en nuestro propio trabajo y transferirlo a más personas, pero sepán que estamos agradecidos.

Más gratitud a las personas que revisaron nuestro borrador y nos brindaron comentarios tan útiles, incluidos Mike Talks, Nikola Sporczyk, Lena Pejgan Wiberg, Pascal Stiefel, Barbara Zaleska, Bertold Kolics, Carol Vaage y nuestra correctora Erica Hunter. Muchas gracias a Constance Hermit por permitirnos usar sus maravillosos dibujos en el Capítulo 12, y a Jenn Sinclair por sus dibujos lineales que usamos tanto.

Un agradecimiento especial a José Díaz y Johanna Rothman por sus encantadores prólogos.

Y por último, pero no menos importante, realmente apreciamos a nuestros esposos, Jack Gregory y Bob Downing, quienes siempre están ahí con una copa de vino cuando la necesitamos.



Prólogo de José Díaz

CEO, Trendig.com

La vida te lleva a tu objetivo. ¿Quién hubiera pensado en 2009, cuando comencé los primeros Agile Testing Days (ATD), que escribiría el prólogo de un libro de Agile Testing Queens, a pesar de que Janet y Lisa han sido parte del conjunto de ATD desde entonces? Estoy muy feliz y honrado de que me hayan pedido que lo haga.

Agile Testing Condensed está en línea con los dos libros anteriores de Janet y Lisa. Esta vez arrojan luz sobre los rincones y exploran los límites de las pruebas y la calidad en proyectos ágiles: breves, concisos, directos y salidos del corazón de dos mujeres que dedicaron sus carreras y su vida a compartir sus conocimientos y hacer avanzar la actividad de probando una profesión ágil. Contiene ejemplos y anécdotas interesantes y es divertido de leer. Ambos autores comparten sus experiencias prácticas e invitan a los lectores a recorrer su camino a través de numerosos proyectos. Te llevan de la mano a dar un paseo por este desafiante y hermoso mundo de las pruebas ágiles y te lo explican. Es un cofre del tesoro, para ti y tu equipo.

Recomiendo no solo este libro inspirador, sino también su capacitación para la vida laboral real "Pruebas ágiles para todo el equipo".

El libro corona su trabajo de los últimos 20 años, en los que han trabajado intensamente con el mundo ágil y su comunidad.

Sus innumerables proyectos, capacitaciones, talleres, conferencias magistrales, charlas, seminarios web, cafés magros, coaching, espacios abiertos, sesiones de tutoría y conversaciones se condensan en este libro y hacen que su contenido sea tan práctico y valioso. Este libro es una lectura obligada para cualquiera que esté considerando emprender un viaje al mundo ágil (incluidos los gerentes).

Está hecho para ti y para mí. ¡Con mucho amor! ¡Disfrutemoslo!

Prólogo de Johanna Rothman

Autor de Cree su proyecto ágil exitoso

Es posible que haya leído otros libros de Janet y Lisa sobre pruebas ágiles.

Esos libros son mucho más que simples pruebas ágiles. Te recomiendo encarecidamente que los leas.

Este libro, Agile Testing Condensed, se centra en cómo crear un entorno en el que el equipo (y especialmente los evaluadores) puedan tener éxito. El libro ofrece muchas fuentes en libros, artículos y publicaciones de blogs que ayudan a reforzar la idea y le ofrecen al lector formas de pensar sobre el tema.

Cada capítulo contiene joyas (esa es la parte condensada) que pueden ayudarle a guiar su pensamiento sobre las pruebas y la calidad.

Por ejemplo, muchos equipos piensan que las pruebas se refieren a cómo un evaluador prueba una característica específica. En cambio, hay una sección que ayuda a todos a pensar en el producto y en cómo planificar el esfuerzo de prueba en los distintos niveles del producto.

Otro ejemplo es cómo nos recuerdan la colaboración. Desde emparejar para explorar hasta trabajar en tríadas para definir historias, los evaluadores colaboran.

Me encantó la sección "Los evaluadores son pegamento de calidad para un equipo". Es cierto, y muy pocos evaluadores y equipos aprovechan ese pegamento.

Si se pregunta cómo ser un tester ágil, o si no está seguro de si necesita testers en su equipo ágil, lea este libro, una lectura breve y rápida que le dará dividendos durante mucho tiempo.

¿Por qué este libro?

Nuestro objetivo con este libro era crear un libro pequeño, conciso y fácil de leer que cualquiera pudiera leer para obtener una comprensión básica sobre cómo tener éxito con las pruebas y construir una cultura de calidad en un contexto ágil. Nuestros dos primeros libros (mucho más extensos) profundizan más y presentan historias de la vida real de profesionales. Nos referimos a esos libros.

como:

- Agile Testing, que es Agile Testing: Una guía práctica para Testers y equipos ágiles, 2009 •
Pruebas más ágiles, que son pruebas más ágiles: aprendizaje Viajes para todo el equipo, 2014

Este libro no es un libro de pruebas introductorio. Hay muchos recursos excelentes disponibles para aprender los conceptos básicos de las pruebas, la automatización de pruebas, DevOps y otros temas. Puedes encontrar algunos buenos enlaces en nuestra bibliografía. De manera similar, esta no es una introducción básica al desarrollo ágil. Es para lectores que forman parte de equipos que adoptan la metodología ágil o para aquellos que desean saber cómo las pruebas y la calidad encajan en el desarrollo ágil y buscan orientación, como gerentes o ejecutivos.

Cómo leer este libro

Puede comenzar en cualquier sección o leer capítulos individuales según lo que desee aprender. Cada capítulo es autosuficiente, aunque podemos referirnos a otros capítulos.

Úsalo como una guía de bolsillo para realizar pruebas ágiles y tenerlo a mano mientras trabaja. Cuando se quede estancado y necesite inspiración, o cuando su equipo se pregunte acerca de un desafío de prueba, consulte este libro en busca de ideas. Cuando quieras profundizar más, consulta nuestros otros libros.

A lo largo del libro encontrará sugerencias que le darán ideas sobre cómo abordar un problema específico.

SECCIÓN 1: Cimentaciones

En esta primera sección, compartimos nuestra definición de "pruebas ágiles".

El corazón de las pruebas ágiles involucra a todo el equipo en las pruebas y la incorporación de calidad a nuestro producto. Hay un gran cambio de mentalidad a medida que el equipo aprende a prevenir defectos en lugar de depender de los evaluadores como red de seguridad para detectarlos.

Los nuevos equipos ágiles aprenden a dividir funciones importantes en partes pequeñas e incrementales y a realizar pequeños cambios con frecuencia. Al mismo tiempo, deben tener presente el panorama general para mantener la satisfacción del cliente.

- Capítulo 1: ¿Qué queremos decir con pruebas ágiles? • Capítulo 2: Enfoque de todo el equipo y mentalidad de pruebas ágiles • Capítulo 3: Planificación de pruebas en contextos ágiles

Capítulo 1: ¿Qué queremos decir con pruebas ágiles?

A lo largo de los años, nos han preguntado muchas veces cómo definir las "pruebas ágiles". Hemos incluido nuestra "definición" de pruebas ágiles en la última parte de este capítulo, pero hay muchos factores que intervienen en esa definición. Algunas de las prácticas que ayudan a apoyar a los equipos en su camino hacia el éxito son:

- Desarrollar la calidad: los equipos se centran en evitar malentendidos sobre el comportamiento de las funciones, así como en prevenir defectos en el código.
- Guiar el desarrollo con ejemplos concretos: utilizando prácticas como el desarrollo basado en pruebas de aceptación (ATDD), el desarrollo basado en el comportamiento (BDD) o la especificación por ejemplo (SBE).
- Incluir actividades de prueba, como tener conversaciones para construir un entendimiento compartido; hacer preguntas para probar ideas y suposiciones; automatizar pruebas; realizar pruebas exploratorias; pruebas de atributos de calidad como rendimiento, confiabilidad y seguridad; y aprender del uso de producción • Usar retrospectivas de todo el equipo y pequeños experimentos para mejorar continuamente las pruebas y la calidad y encontrar lo que funciona en su contexto.

Ampliaremos las prácticas anteriores a lo largo de este libro. No las consideramos "mejores prácticas" porque sabemos que están en constante evolución.

Modelos de prueba continua

Las pruebas son una parte integral del desarrollo de software, junto con la codificación, las operaciones, la comprensión de las necesidades del cliente y más. Nos gustan los modelos que representan un enfoque de prueba continuo u holístico. Uno de nuestros favoritos es el enfoque que utilizan Ellen Gottesdiener y Mary Gorman en su libro Discover to Deliver, 2012.

Como se muestra en la Figura 1.1, el proceso de desarrollo representa un bucle infinito, que se confirma continuamente: cuál es la forma en que realmente desarrollamos software. Aprendemos sobre una característica que nuestros clientes desean, la creamos y la entregamos, y luego aprendemos cómo la usan realmente los clientes. Usamos esos comentarios para decidir qué construir (o eliminar) próximo.

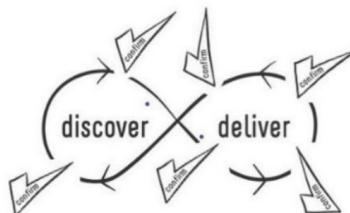


Figura 1.1: Circuito Descubrir para Entregar © 2015 por EBG Consulting

Dan Ashby adoptó un enfoque similar utilizando el siguiente diagrama (Figura 1.2) en su publicación de blog mientras habla sobre [las pruebas en DevOps](#)¹.

Este modelo muestra que las pruebas son una parte integral de DevOps. Entraremos en más detalles sobre este tema en el Capítulo 8.

¹<https://danashby.co.uk/2016/10/19/continuous-testing-in-devops/>



Figura 1.2: Pruebas continuas en el bucle DevOps

Nos gustan términos como prueba continua o prueba holística y reconocemos que cada equipo necesita adaptarlos a sus necesidades únicas. contexto.

Diez principios para pruebas ágiles

En Agile Testing, presentamos la idea de 10 principios para Agile Testers. Ahora nos damos cuenta de que estos principios no son necesariamente solo para los evaluadores sino para cualquier miembro del equipo. Escribimos estos principios en un momento en el que la mayoría de los evaluadores todavía formaban parte de un equipo de pruebas con un proyecto por fases y cerrado, y las pruebas estaban al final, después de que toda la codificación estaba "terminada".

Estos principios todavía se aplican hoy en día para cualquier persona en un equipo ágil que desee ofrecer el producto de la más alta calidad posible.

- Proporcionar retroalimentación continua.
- Entregar valor al cliente. • Permitir la comunicación cara a cara.
- Ten coraje. • Manténlo simple.

- Practicar la mejora continua.
- Responder al cambio.
- Autoorganizarse.
- Centrarse en las personas.
- ¡Disfrutar!

Manifiesto de prueba

Terminaremos este capítulo con este “[manifiesto de prueba](#)²” creado por Karen Greaves y Samantha Laing. Su manifiesto refleja el cambio de mentalidad necesario para un enfoque de pruebas ágil exitoso. Realizamos pruebas durante todo el proceso de desarrollo, nos centramos en prevenir errores, probamos mucho más que la funcionalidad y todo el equipo asume la responsabilidad de la calidad. Encontrará estos principios reflejados en todos nuestros libros. Cada vez que su equipo esté atrapado en un problema de calidad o de prueba, reflexione sobre estos principios para encontrar una manera de seguir adelante. (Figura 1.3).



Figura 1.3: El manifiesto de prueba

²<http://www.growingagile.co.nz/2015/04/the-testing-manifesto/>

Definición de pruebas ágiles

La definición más simple que se nos ocurrió de lo que entendemos por pruebas ágiles es la siguiente:

Prácticas de prueba colaborativas que ocurren continuamente, desde el inicio hasta la entrega y más allá, respaldando la entrega frecuente de valor para nuestros clientes. Las actividades de prueba se centran en incorporar calidad al producto, utilizando ciclos de retroalimentación rápidos para validar nuestra comprensión. Las prácticas fortalecen y apoyan la idea de la responsabilidad de todo el equipo por la calidad.

Lleva un tiempo digerirlo y puedes consultar nuestra [publicación de blog³](#) para más detalles.

³<https://agiletester.ca/ever-evolving-never-set-stone-definition-agile-testing/>

Capítulo 2: Todo el equipo

Enfoque y Ágil

Mentalidad de prueba

Muchos equipos de software todavía utilizan un enfoque lineal por fases para entregar software. Las personas que desempeñan un rol determinado están aisladas en equipos específicos y pasan el trabajo de un equipo a otro. Se considera que el equipo de pruebas o control de calidad es responsable de garantizar la calidad, generalmente al final del proceso y justo antes de la entrega a producción, cuando ya es demasiado tarde para hacer mucho para mejorar la calidad.

En el desarrollo ágil, derribamos los silos y convertimos el desarrollo en un proceso continuo e iterativo. Todo el equipo de entrega trabaja en conjunto para generar calidad durante todo el proceso.

Por "todo el equipo", normalmente nos referimos al equipo de entrega: las personas responsables de comprender qué construir, construirlo y entregar el producto final al cliente.

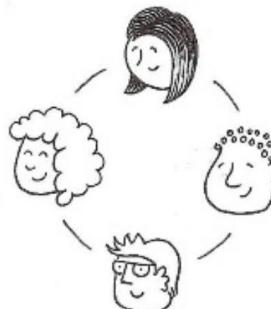


Figura 2.1: Un solo equipo

En organizaciones más grandes, incluso aquellas que han adoptado principios y prácticas ágiles, puede haber más de un equipo trabajando en un producto, como un equipo de base de datos independiente, un equipo de experiencia del usuario u otro equipo de producto. En estos casos, la definición de equipo completo se extiende a cualquier persona que necesite para entregar el producto. El movimiento DevOps ha hecho más visible la inclusión de operaciones en la entrega. Janet se refiere a las personas ajena al equipo de parte como una familia extendida.

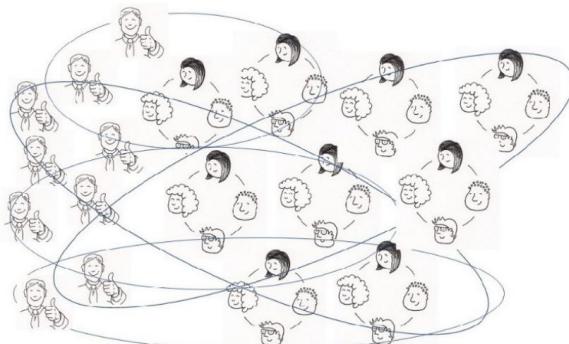


Figura 2.2: Múltiples equipos

Centrarse en la calidad

El enfoque de todo el equipo significa que todos los miembros del equipo son responsables de la calidad de su producto. Parte de esta responsabilidad es garantizar que las tareas de prueba se completen junto con el resto de las tareas de desarrollo. Cuando el objetivo es ofrecer la mayor calidad posible, en lugar de hacerlo más rápido, el equipo construye una base sólida de prácticas. Para lograr ese nivel de calidad, los equipos administran su carga de trabajo para tener tiempo de aprender prácticas básicas como el desarrollo basado en pruebas (TDD) y las pruebas exploratorias. También se toman tiempo para aprender el dominio empresarial y establecer relaciones con expertos empresariales para identificar las características más comerciales.

valor y luego implementarlos de la manera más simple posible. Con el tiempo, al centrarse en la calidad, los equipos empiezan a poder trabajar más rápido.

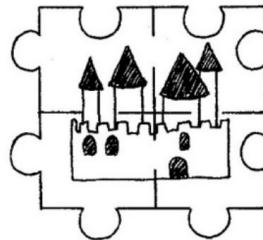


Figura 2.3: El valor de su negocio entregado según lo esperado

Hay varias áreas que requieren un cambio en la forma en que los miembros del equipo abordan el desarrollo. Cuando todo el equipo es responsable de la calidad del producto así como de la calidad del proceso, cada miembro del equipo debe ser proactivo en la resolución de problemas. Por ejemplo, todos los miembros del equipo pueden ayudar a descubrir qué es más valioso para los clientes. Trabajan para ofrecer la cantidad suficiente de ese valor en pequeños incrementos para aprender cómo el cliente utiliza esa capacidad.

Al crear estos ciclos de retroalimentación rápida, el equipo puede centrar sus pruebas en las funciones que son más valiosas para el cliente.

Cada equipo debe discutir y acordar una “valiosa Definición de Hecho” (DoD). Eso debería incluir cómo el equipo planea lidiar con los defectos encontrados en el código. El Departamento de Defensa debe incluir pruebas, y la pregunta que debe hacerse es: “¿A qué tipos de pruebas te refieres?” En el Capítulo 9, cubrimos los cuadrantes de pruebas ágiles y respondemos esa pregunta. Todos los miembros del equipo deben entender el Departamento de Defensa de la misma manera.

Cómo los equipos abordan los defectos

Un gran cambio de mentalidad para los evaluadores es adoptar un enfoque de equipo para solucionar defectos. En More Agile Testing, hablamos de centrarnos en la pre-

ventilar defectos en lugar de encontrarlos una vez completada la codificación.

Cuando todo el equipo se concentra en incorporar calidad al producto mediante el uso de prácticas como el desarrollo basado en pruebas de aceptación y pensar en las limitaciones en torno a los atributos de calidad, se puede contribuir en gran medida a reducir la cantidad de defectos encontrados en el código.

Muchos equipos practican la tolerancia cero a los defectos. Esto significa que ningún defecto conocido escapa a una iteración o a la finalización de la historia. Para que esto funcione, los equipos deben recibir retroalimentación rápida de las actividades de prueba para que cualquier defecto encontrado pueda solucionarse de inmediato. Una vez encontrado, los programadores escriben una o más pruebas ejecutables, corrigen el código para que las pruebas pasen y realizan pruebas exploratorias si es necesario. El equipo puede entonces olvidarse de ello, sabiendo que han corregido el asunto.

Muy a menudo, este cambio filosófico de enfoque ayuda a transformar un entorno adversario en un entorno cooperativo.

Multiples perspectivas

Los miembros del equipo tienen diferentes puntos de vista, habilidades y perspectivas. Descubrimos que al utilizar todas las perspectivas, comprendemos mejor los riesgos involucrados al ofrecer una función. Por ejemplo, diseñar para la capacidad de prueba ayuda a convertir ejemplos de comportamiento de software deseado y no deseado en pruebas ejecutables. Los miembros del equipo se convierten en especialistas generalizados, es decir, pueden ser expertos en una o dos áreas pero pueden contribuir a los objetivos comunes del equipo de diversas maneras.

Algunos ejemplos:

- Los evaluadores pueden ser expertos en probar el producto, pero pueden contribuir a comprender las características y las historias haciendo preguntas para descubrir suposiciones ocultas.
- Los programadores pueden realizar pruebas exploratorias por su cuenta. código antes de registrar su código.

- Los propietarios de productos ejecutan pruebas de aceptación en cada historia.

En el Capítulo 11, hablaremos un poco más sobre el rol de un evaluador y cómo puede cambiar para los equipos ágiles.

Capítulo 3: Planificación de pruebas en contextos ágiles

Uno de nuestros siete principales factores de éxito de Agile Testing es "No olvide el panorama general". Los equipos a menudo quedan atrapados en la creación, prueba y entrega de pequeños incrementos (lo cual recomendamos) y se olvidan de cómo encaja esa pequeña porción en el sistema o cómo funciona para resolver el problema empresarial.

Para planificar las actividades de prueba de manera efectiva, un equipo debe considerar su contexto. Para comprender su contexto, piense en tres aspectos del mismo: el equipo, el producto y los niveles de detalle de su sistema.

El equipo

No todos los equipos son iguales. Si forma parte de un equipo pequeño que comparte ubicación, tiene una situación ideal para facilitar la comunicación. Tienen muchas posibilidades de aprender los valores de los demás y compartirlos. Es un punto óptimo para ofrecer un gran producto y la planificación es mucho más fácil. Los equipos pueden comprender fácilmente la siguiente característica y profundizar en la planificación a nivel de historia y tareas.

Sin embargo, muchas personas trabajan en organizaciones grandes y distribuidas globalmente. Eso trae consigo diferentes desafíos. Las organizaciones más grandes tienen múltiples proyectos y muchos equipos. Cuando adoptan la metodología ágil, a menudo reemplazan los silos basados en roles, como desarrolladores y control de calidad, con Scrum o silos de equipos de funciones.

Cuando muchos equipos grandes trabajan en la misma base de código, la integración puede convertirse en un gran desafío. Los equipos pueden necesitar especialistas como evaluadores de rendimiento, seguridad y confiabilidad, pero es posible que no haya

suficientes para distribuirlos entre todos los equipos multifuncionales. Es incluso difícil hacer visibles todos estos problemas y desafíos. La planificación a nivel de lanzamiento es particularmente desafiante en este entorno, pero es fundamental para ofrecer nuevas capacidades a los clientes.

No importa cuál sea el contexto, el equipo de ejecución debe asumir la responsabilidad de planificar y completar todas las actividades de prueba, incluso si eso significa contratar especialistas. Si tienen dependencias, deben trabajar con otros equipos para gestionar o eliminar esas dependencias, preferiblemente antes de que comience la codificación. Dicho esto, es necesario hacer ajustes para adaptarse a cada contexto único.

El producto

El nivel de calidad que desean las partes interesadas depende de su producto, así como del tipo y cantidad de pruebas que puedan ser necesarias. Por ejemplo, un sistema de gestión de contenidos utilizado únicamente por usuarios internos tiene prioridades diferentes a las del software de dispositivos médicos.

Cada uno tiene un entorno diferente en el que habita e implica diferentes riesgos.

Considere el tamaño de su producto, cuántas personas lo usan o si está integrado con aplicaciones externas. Piense en cómo se entrega el producto y el riesgo asociado con el mecanismo de entrega. Por ejemplo, si la organización aloja su propia aplicación web, tiene mucho más control sobre cuándo y con qué frecuencia se actualiza el producto. O si el producto necesita funcionar en muchos dispositivos, como teléfonos, ¿cómo se realizan las actualizaciones sin interrumpir el uso habitual?

Uno de los principales objetivos de las pruebas es identificar y mitigar los riesgos, tanto para el usuario como para la empresa. Obviamente, esto juega un papel importante en la forma en que planifica sus pruebas. Ésta es una de las razones por las que los equipos de entrega necesitan aprender el ámbito empresarial y trabajar en estrecha colaboración con expertos empresariales. La experiencia en el dominio ayuda a la hora de planificar qué probar. ¿Tiene su equipo una muy buena idea de cómo

¿Se utiliza el producto? ¿Todos los miembros del equipo tienen conocimiento del dominio? Colaborar con expertos en productos y negocios ayuda al equipo de entrega a encontrar formas óptimas de desarrollar capacidades que sus clientes valoren.

Hay muchas cosas a considerar en torno al dominio de su producto. No es sólo el software lo que estás probando; es el producto del que dependen sus usuarios finales.

Planificación a través de niveles de detalle

Las pruebas en múltiples niveles (Figura 3.1) requieren una planificación adicional. Los ciclos de lanzamiento generalmente comienzan determinando lo que se podría entregar en la primera "versión de aprendizaje". Quizás solo se entregue parte de una función. Las funciones se dividen en historias y se priorizan para que el equipo sepa cuál entregar primero. Es importante que el equipo comprenda el panorama general antes de incorporar historias a una iteración. Cuando los desarrolladores trabajan en una historia, se concentran más en asegurarse de que se completen las tareas individuales. Los evaluadores a veces caen en la trampa de pensar sólo en la historia que están probando, por lo que los recordatorios del panorama general son importantes.

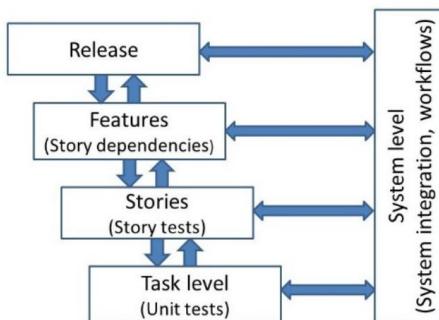


Figura 3.1: Niveles de detalle para la planificación

Nivel de versión/función

Los equipos deben comprender cómo cada historia entregada puede afectar el panorama general, especialmente en organizaciones más grandes o globales. Cada versión podría estar compuesta por muchas características, que a su vez pueden estar compuestas por muchas historias y tareas que tienen un impacto en el sistema en su conjunto.

En organizaciones grandes con varios equipos trabajando en el mismo producto, uno de los problemas que vemos a menudo es que los equipos tienden a "aislarse". Se olvidan de hablar con otros equipos para solucionar posibles dependencias.

La Figura 3.2 muestra la importancia de un enfoque de prueba que incluya a todos los equipos trabajando para lograr el lanzamiento de un único producto. Para brindar una visión general de la cobertura de las pruebas, considere reunir personas de diferentes equipos para crear un mapa mental de pruebas o una matriz de pruebas de funciones (detalles en Pruebas más ágiles) que abarque el producto.

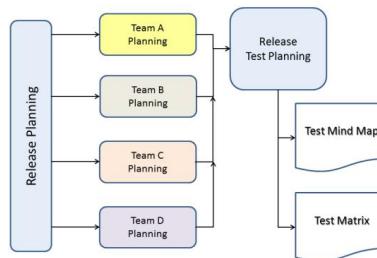


Figura 3.2: Planificación para múltiples equipos

Recuerde, una talla única no sirve para todos, así que tenga en cuenta el tamaño y la cantidad de sus equipos, dónde están ubicados, cómo se coordinará el trabajo entre los equipos y si todas las habilidades necesarias para las pruebas están disponibles para cada equipo.

Idealmente, las actividades se coordinan con otros equipos a medida que avanza el desarrollo de funciones. Sin embargo, es importante tener en cuenta que es posible que se necesite un producto más finalizado para cosas como agregar capturas de pantalla finales a la documentación de usuario o de capacitación. O porque no lo es

Por lo general, implementar un parche para una aplicación móvil es un proceso rápido; es posible que se necesiten pruebas adicionales en las que todo el equipo explora la versión más reciente una vez más.

Sugerencia: no cometa el error de hacer que los evaluadores y quizás el personal de operaciones finalicen el final previo a la implementación mientras los desarrolladores comienzan nuevas historias. Al igual que el desarrollo, la preparación para la entrega debe ser un esfuerzo de todo el equipo.

nivel de historia

En este nivel, no importa si los equipos limitan el tiempo de sus iteraciones o trabajan con un método basado en flujo como Kanban.

Comience con pruebas de aceptación de alto nivel (consulte el Capítulo 4: Guía del desarrollo con ejemplos para obtener más detalles). Obtenga ejemplos para aumentar la comprensión compartida de la historia y convierta esos ejemplos en pruebas. Si las pruebas se escriben antes de que se realice la codificación, ayudan a guiar el desarrollo y prevenir defectos.

Sugerencia: considere qué cartas de pruebas exploratorias podrían ser necesarias (consulte el Capítulo 6). Piense en las limitaciones del producto y lo que eso significa para probar los atributos de calidad (Capítulo 7).

A medida que los equipos planifican las pruebas y discuten la implementación de cada historia, surgen detalles sobre las pruebas. Crear nuevos ejemplos y pruebas para reflejar lo aprendido sobre la historia.

Nivel de tarea

Los programadores utilizan el desarrollo basado en pruebas (TDD) para escribir pruebas de bajo nivel (unitarias) antes de cada pequeño fragmento de código. Algunos programadores lo llaman desarrollo impulsado por el diseño porque les ayuda a

diseñar su código para que sea comprobable. Estas pruebas son relativamente fáciles de escribir, se ejecutan rápidamente y brindan retroalimentación rápida al equipo. Forman gran parte de la base del modelo piramidal de automatización de pruebas que analizamos en el Capítulo 10: Visualización de una estrategia de automatización de pruebas.

Planificación de pruebas de regresión

Las pruebas de regresión consisten en asegurarse de que el sistema haga lo que hizo ayer. Las prácticas contemporáneas para realizar pequeños cambios en la producción frecuentemente no dejan tiempo para realizar una verificación de regresión manual completa, por lo que la automatización de las pruebas se crea a medida que se desarrolla el producto. La automatización de pruebas debe ser parte de cada historia, especialmente en la capa de servicio (ver Capítulo 10). Si una historia cambia de funcionalidad, asegúrese de incluir tareas para cambiar las pruebas existentes.

Las pruebas de regresión automatizadas nos permiten tener confianza en nuestro producto con comentarios rápidos. Muchas (si no todas) las pruebas se ejecutan como parte de la integración continua (CI). Algunos equipos programan pruebas más lentas para ejecutarlas con menos frecuencia. Por ejemplo, ejecutarlos varias veces al día en lugar de cada compilación. El uso de funciones de lanzamiento para “ocultar” cambios a los usuarios de producción permite a los equipos realizar algunas actividades de prueba de forma asincrónica a medida que implementan continuamente en producción. La función se “activa” en producción cuando se completan todas las pruebas (consulte el Capítulo 8: Pruebas en DevOps).

Encontrará más información sobre esto en el Capítulo 23: Pruebas y DevOps en pruebas más ágiles.

SECCIÓN 2: Pruebas Enfoques

En esta sección, profundizamos en las prácticas básicas para pruebas ágiles. Utilizar ejemplos concretos para guiar el desarrollo es una de las formas más efectivas de generar confianza en cada nuevo cambio. Compartimos formas de ayudar a los miembros del equipo en diferentes roles a aprender a colaborar para incorporar calidad al producto. Las pruebas exploratorias son otra práctica fundamental para generar confianza en la que todo el equipo debe participar.

Los equipos ágiles a menudo caen en la trampa de probar únicamente la funcionalidad: cómo debe comportarse cada característica o capacidad. Hay muchos otros atributos de calidad importantes que debemos probar y que también cubrimos en esta sección.

DevOps y la entrega continua son temas candentes en la actualidad. Observamos cómo las pruebas y los evaluadores encajan y ayudan a su equipo a tener éxito con esos enfoques.

- Capítulo 4: Guiar el desarrollo con ejemplos • Capítulo 5: Habilitar la colaboración • Capítulo 6: Pruebas exploratorias • Capítulo 7: Prueba de atributos de calidad • Capítulo 8: Pruebas en DevOps

Capítulo 4: Guiando Desarrollo con Ejemplos

Muchos equipos han utilizado durante años la idea de utilizar ejemplos para guiar el desarrollo de funciones e historias. Lo vemos como un enfoque valioso y probado. Los profesionales líderes continúan encontrando nuevas formas de ayudar a los equipos a tener éxito con estas técnicas: por ejemplo, el mapeo de ejemplo de Matt Wynne (ver más en el Capítulo 5).

Ejemplos concretos de comportamiento del sistema deseado y no deseado ayudan a los equipos a desarrollar una comprensión compartida de cada característica e historia. Esto les permite crear lo correcto con menos rechazos de historias y un ciclo más corto desde el inicio hasta la implementación de producción. Los evaluadores contribuyen solicitando estos ejemplos concretos y utilizándolos para crear pruebas ejecutables que guíen el desarrollo. Pueden ser la voz de la experiencia al liderar estas conversaciones.

Métodos basados en ejemplos

Existen algunas variaciones para crear funciones e historias basadas en ejemplos. El desarrollo impulsado por el comportamiento (BDD) es el que Janet escucha que la mayoría de los equipos afirman que utilizan. BDD, presentado por primera vez por Daniel Terhorst-North, captura escenarios de ejemplo en un lenguaje natural específico del dominio. La sintaxis “Dado/Cuando/Entonces” describe condiciones previas, algunas acciones desencadenantes y la condición posterior resultante. A medida que los desarrolladores escriben el código de producción, es probable que automaticen algunos o todos los escenarios para ayudar a saber cuándo han entregado lo que el cliente desea.

Escribir estos escenarios puede parecer fácil, pero se necesita práctica para simplificar las pruebas de modo que realmente solo se esté probando una cosa. Consulte la Figura 4.1 para ver un ejemplo.

El desarrollo basado en pruebas de aceptación (ATDD) es similar. Es una forma más genérica de guiar el desarrollo con ejemplos sin lenguaje ni reglas estrictas. La mayoría de la gente utiliza ATDD para pruebas funcionales, aunque se pueden incluir requisitos para otros atributos de calidad, como seguridad o accesibilidad. Un enfoque ATDD que hemos utilizado es capturar al menos un ejemplo de alto nivel de comportamiento deseado o de “camino feliz” y al menos un ejemplo para cada tipo de mal comportamiento mientras el equipo planifica la historia. Los evaluadores y otros miembros del equipo obtienen ejemplos más detallados a medida que avanza la codificación de la historia. Al menos algunos de los ejemplos se convierten en pruebas ejecutables que ayudan al equipo a decidir cuándo han terminado.

La Figura 4.1 muestra una progresión que comienza con la división de la función en historias. Las burbujas azules se crean como parte de talleres de preparación de historias, refinamiento del trabajo pendiente o discusiones de “tres amigos”.

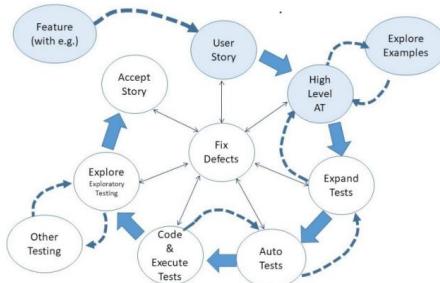


Figura 4.1: Desarrollo basado en pruebas de aceptación (ATDD)

Los equipos que practican la Especificación por ejemplo (SBE) comienzan identificando objetivos en torno a la historia utilizando un enfoque como el mapeo de impacto (ver más en el Capítulo 5). Luego, el equipo obtiene ejemplos clave, que se convierten en las especificaciones, durante un taller de especificaciones. A medida que avanza el desarrollo, los ejemplos se refinan y se convierten en

especificaciones ejecutables para validar el producto con frecuencia. Estos ejemplos ejecutables, al igual que BDD y ATDD, se convierten en documentación viva de la aplicación. Cuando Gojko Adzic acuñó el término Especificación mediante el ejemplo, deliberadamente no utilizó la palabra "prueba" para describir ninguna de las actividades.

Por qué los ejemplos ayudan

Observar cada nueva capacidad desde una variedad de perspectivas ayuda a que sea más probable que un equipo identifique el valor que los clientes obtienen de cada nueva capacidad. Esta diversidad ayuda a cada miembro del equipo a superar prejuicios inconscientes y a "pensar fuera de lo común".

Cuando a cada parte interesada se le piden ejemplos concretos del comportamiento del sistema, los equipos analizan esos ejemplos y es fácil ver discrepancias. También es más fácil profundizar hasta el valor mínimo específico de lo que los clientes necesitan y permite a los equipos entregar "lo suficiente" de lo correcto.

Le mostraremos lo que queremos decir usando un escenario de ejemplo.

Historia: Como comprador canadiense, quiero darle efectivo al cajero y espero el cambio correcto para pagar solo la cantidad correcta por mis compras. La caja registradora informa la cantidad correcta de cambio a dar.

Escenario: El camino feliz donde el comprador da más de la cantidad de la compra y recibe el cambio correcto.

Dado que soy un comprador canadiense y he comprado
alimentos por valor de \$

4,97, cuando le doy al cajero \$

5,00, espero recibir \$ 0,05 de cambio.

Nota: Al usar este ejemplo, una regla comercial que el equipo quizás no haya considerado es que en Canadá no hay centavos en uso,

entonces el número se redondea hacia arriba o hacia abajo y el cambio se da a la moneda de cinco centavos más cercana (0,05).

Una de las formas favoritas de Janet de mostrar ejemplos es en formato tabular. Muestra rápidamente lo que se está perdiendo y lo que la gente piensa mientras mantienen la discusión. Cada línea puede convertirse en una prueba. La Figura 4.2 muestra este formato para el escenario que utilizamos anteriormente.

Grocery total	Cash given	Change returned	Test
4.97	5.00	0.05	Happy path
4.97	4.00	0.00	Not enough cash
4.97	10.00	5.05	Happy path

Figura 4.2: Ejemplo de formato tabular utilizando ejemplos concretos

Hay muchas buenas maneras de estructurar conversaciones en las que los equipos puedan obtener ejemplos. [Mapeo de la historia](#) de Jeff Patton ayuda a todo el equipo a recorrer el recorrido del usuario a través de su producto. Nos gusta capturar reglas comerciales específicas, así como ejemplos que las ilustran con mapas de ejemplo. Conversaciones estructuradas utilizando [las 7 dimensiones del producto](#) de Ellen Gottesdiener y Mary Gorman. Asegúrese de que los equipos obtengan ejemplos de muchos aspectos diferentes de valor.

Vale la pena probar cualquier técnica que promueva la colaboración cara a cara en pequeños grupos multifuncionales. Consulte el Capítulo 7 en Pruebas más ágiles para obtener más detalles e historias.

Esta es tu base

Reunir a los miembros del equipo de entrega y de negocios para recopilar ejemplos es clave para entregar valor a los clientes a un ritmo frecuente y sostenible. Permite a los equipos desarrollar el entendimiento compartido más importante antes de comenzar a codificar. Ayuda a todos

<https://www.jpattonassociates.com/user-story-mapping/>

<https://discovertodeliver.com/image/data/Resources/visuals/DtoD-7-Product-Dimensions.pdf>

permanecer anclado en la realidad. Los ejemplos concretos se convierten en pruebas ejecutables que garantizan que construimos lo correcto y que seguirá funcionando correctamente en el futuro hasta que los clientes quieran cambiar él.

Sugerencia: cuando se encuentre en una discusión agitada sobre los requisitos de una característica o una discusión sobre cómo debe comportarse una determinada capacidad, DETÉNGASE. Pide un ejemplo. Aún mejor, haga que el grupo comience a dibujar ejemplos en la pizarra (reales o virtuales). Ahorrará mucho tiempo y estará más cerca de construir lo correcto.

Capítulo 5: Habilitación Colaboración

La colaboración dentro de un equipo y entre equipos es una de las piedras angulares que hacen que los equipos ágiles tengan éxito. Sin embargo, descubrimos que muchos equipos no tienen idea de cómo empezar a construir esas relaciones. En este capítulo, hablaremos sobre algunas prácticas muy simples que pueden ayudarlo a usted y a su equipo a obtener tracción.

Colaborar con los clientes

Comencemos colaborando con el cliente, normalmente representado por el propietario del producto. Si los equipos no entienden qué problema está tratando de resolver el cliente, es posible que resuelvan el problema equivocado. Es esencial que los equipos trabajen con sus clientes para comprender sus verdaderas necesidades.

En primer lugar, recomendamos encarecidamente que todos los miembros del equipo comprendan el dominio. Esto se puede lograr trabajando estrechamente con los usuarios finales, pidiendo ejemplos o escenarios, o incluso haciendo dibujos en pizarras para comprender las diferencias y aclarar significados.

Preguntas como estas harán que el cliente considere el uso y el riesgo asociado.

“¿Cómo usarás esto?”

“¿Qué es lo peor que puede pasar?”

Los evaluadores pueden facilitar la comunicación entre el desarrollador y el cliente, pero es importante no estorbar. Llamamos a la práctica de conseguir un

tester, un programador y un experto en negocios (propietario de producto, gerente de producto o analista de negocios) juntos para hablar sobre una historia de usuario, el "Poder de tres". George Dinwiddie se refiere a ellos como los [Tres Amigos](#). Es una forma poderosa de generar un entendimiento compartido sobre historias, características y cómo encajan en el producto.

Sugerencia: reúna al evaluador, al programador y al experto en negocios, y tal vez a uno o dos roles más, cada vez que surja una pregunta. Por ejemplo, una pareja de desarrolladores está trabajando en una nueva historia y una de las pruebas comerciales falla. Van a hablar con un evaluador y le dicen que creen que la prueba espera un comportamiento incorrecto. Ese es el momento de buscar un propietario de producto y tener una discusión tripartita. Esta rápida conversación ahorra mucho tiempo después al intentar corregir un defecto que apareció en el código.

Dependiendo del producto y del tipo de funciones que se estén desarrollando, es posible que se necesiten más perspectivas, como las de un diseñador de UX, un experto en datos o un experto en operaciones.



Figura 5.1: Invitar a las personas adecuadas

Mapeo de impacto

Marcos como [el mapeo de impacto](#) son útiles para decidir qué funciones debemos crear y tal vez incluso determinar cuál debería ser la prioridad. Comience con el objetivo de una función (el "por qué"). Entonces

<https://www.agileconnection.com/article/tres-amigos-strategy-developing-user-stories>
<https://www.impactmapping.org/>

identificar quién podría ayudarnos a lograr ese objetivo y quién podría interponerse en nuestro camino. Para cada “quién”, pregunte cómo podrían ayudarnos o dificultarnos el logro de la meta (esos son los impactos). Por último, piense en los resultados que podrían derivarse de cada impacto (el “qué”). Este ejercicio ayuda al equipo a comprender el panorama general y las razones detrás de lo que están desarrollando.

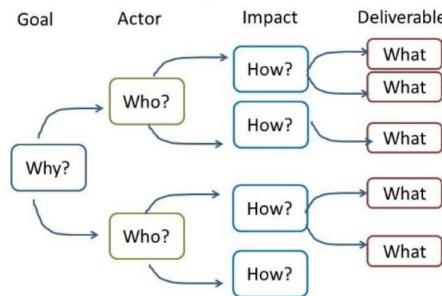


Figura 5.2: Mapeo de impacto

Responde a la pregunta "¿Cómo sabremos si esta función logra el objetivo después de que la lancemos?"

Hacer preguntas

Es común que una reunión de planificación de funciones comience con una discusión sobre cómo implementar la función. A veces, el propietario del producto ha tenido sus propias ideas: "Tome el mismo código que usamos para los códigos de descuento y conviértalos en cantidades negativas para que podamos agregar recargos". (Sí, Lisa tuvo exactamente esa experiencia). Es importante no permitir que eso suceda: comience con el por qué.

Cuando se reúne con una parte interesada de la empresa, como el propietario de un producto, para hablar sobre las próximas funciones, la primera pregunta que debe hacerse es: "¿Por qué estamos implementando esta función?" Otras buenas preguntas: "¿Qué problema resolverá esto para la empresa, el cliente o el usuario final?"

QA significa "Question Asker", una idea que obtuvimos de Pete Walen.

Los evaluadores rutinariamente hacen preguntas que nadie más piensa en hacer, por lo que si no tiene un evaluador en el equipo, intente designar un rol de formulador de preguntas.

Los equipos experimentados suelen incorporar criterios de calidad en su forma de trabajar. Por ejemplo, si quieren evitar problemas de seguridad como secuencias de comandos entre sitios (XSS) e inyección SQL, probablemente estén integrados en la arquitectura del sistema.

Sin embargo, según nuestra experiencia, las partes interesadas del negocio a menudo suponen incorrectamente que el equipo técnico ya sabe qué atributos de calidad son importantes: atributos como cuántos usuarios simultáneos utilizarán el producto, qué dispositivos deben ser compatibles o qué tan rápido es el tiempo de respuesta percibido de los usuarios. una aplicación debe ser. Consulte el Capítulo 7 para obtener más información sobre los atributos de calidad.

Hacer preguntas específicas y abiertas ayuda a exponer suposiciones ocultas.

- “¿Es posible que podamos implementar esta característica y no resolver el problema?”
- “¿Qué harán los usuarios antes de utilizar esta función?” • “¿Qué harán después?”

Mapeo de ejemplo

Matt Wynne nos presentó la idea del [mapeo de ejemplo](#), y descubrimos que es una excelente manera de explorar una nueva función y el valor que debería ofrecer. Trabaje con el propietario del producto u otras partes interesadas sobre las reglas comerciales en una discusión tipo "Poder de tres".

Las reglas comerciales son excelentes lugares para comenzar a explorar una característica, ya que pueden ayudarnos a dividir una característica en historias, así como a obtener una comprensión compartida de cómo debe comportarse la característica.

<https://cucumber.io/blog/example-mapping-introduction/>

La técnica de mapeo de ejemplos de Matt Wynne es una base muy eficaz para este tipo de conversación porque se utilizan ejemplos concretos para ayudar a aclarar nuestra comprensión de las reglas. A medida que continúa la conversación, tenga presente el objetivo principal y concéntrese en el valor que la función ofrece a los clientes y usuarios finales. Los equipos a menudo descubren que se exponen más reglas comerciales como resultado del uso de ejemplos reales.

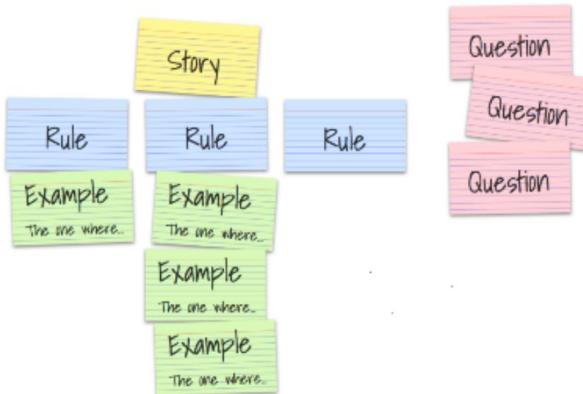


Figura 5.3: Ejemplo de mapeo

Utilizar un mapeo de ejemplos para obtener reglas de negocio, ejemplos y preguntas que necesitan respuesta es una forma efectiva de asegurarse de que todo el equipo comience en la misma página cuando planifican la iteración.

Genere confianza utilizando la visibilidad

Las pruebas permiten a los equipos identificar riesgos para que los clientes puedan tomar las mejores decisiones, lo que a su vez genera confianza. Cuando el equipo solicita su opinión, los clientes tienen la confianza de que obtendrán un software que funcione.

Podemos utilizar mapas mentales, diagramas de flujo, diagramas de contexto, estados.

diagramas u otras herramientas para observar las dependencias y los efectos dominó de cada nueva característica. Si nuestras funciones no son fácilmente entendidas y utilizadas por todos, no pueden proporcionar el valor deseado. Estar atento al panorama general es una de las fortalezas que los evaluadores aportan a la metodología ágil equipos.

Dibujar en una pizarra mientras se habla de una historia es una forma comprobada de optimizar la comunicación. Levantarse y moverse ayuda a las personas a pensar y aprender.

Pista: Coge un marcador y algunas tarjetas, pegatinas y una pizarra.

Haga que las personas se pongan de pie y participen activamente dibujando o moviendo fichas o notas adhesivas.



Figura 5.4: Utilice la visibilidad para generar confianza

Si hay participantes remotos, utilice herramientas colaborativas en línea para ayudar. Nuestros equipos han descubierto que el uso de mapas mentales, ya sea en una pizarra física o mediante una herramienta colaborativa en tiempo real como Mindmup, puede ser extremadamente eficaz para ayudar a identificar las incógnitas y encontrar soluciones creativas.

Este tipo de ayudas visuales permiten a los equipos hacer preguntas mejores y más específicas. Cada vez que el equipo encuentra un problema, encuentra una manera de hacerlo visible, de modo que puedan empezar a pensar en experimentos para reducir el problema.

Capítulo 6: Explorar Continuamente

Los equipos más ágiles están encontrando valor en las pruebas exploratorias, pero sigue siendo una idea nueva o desconocida para muchos equipos. Comenzaremos explicando el propósito de las pruebas exploratorias y lo que implica.

En Explore It!, Elisabeth Hendrickson define las pruebas exploratorias como "... diseñar y ejecutar pruebas simultáneamente para aprender sobre el sistema, utilizando los conocimientos del último experimento para informar el siguiente".

En las pruebas exploratorias, una persona interactúa con el sistema y observa el comportamiento real, diseñando pequeños experimentos. En función de lo que aprenden, adaptan el experimento y continúan aprendiendo más sobre el sistema. En el proceso, pueden hacer descubrimientos sorprendentes, incluidas implicaciones de interacciones que nadie había considerado. Las pruebas exploratorias exponen malentendidos sobre lo que se supone que debe hacer el software.

Los evaluadores, programadores u otros miembros del equipo que realizan pruebas exploratorias deben estar abiertos a observar, aprender, utilizar habilidades de pensamiento crítico y desafiar las expectativas. El objetivo de las pruebas exploratorias es reducir el riesgo y ganar confianza en el producto. No hay scripts ni listas de resultados esperados. En cambio, se identifica una meta, con recursos (o variaciones) y una misión. Los miembros del equipo toman notas mientras exploran, aprenden y luego informan a otros miembros del equipo y a las partes interesadas del negocio. Como resultado de un

En la sesión de prueba exploratoria, el evaluador puede mostrar cualquier error encontrado a sus compañeros de equipo o proponer nuevas características o historias que puedan ser necesarias.

Las pruebas exploratorias son un enfoque disciplinado de las pruebas. No debe confundirse con las pruebas ad hoc, que se realizan sin ningún

planificación o documentación, o pruebas con monos, que implican ingresar entradas aleatorias y acciones aleatorias para ver qué falla. Piense en la diferencia entre deambular al azar (quizás perdido) y explorar reflexivamente (para obtener información y con un propósito).

Presentaremos algunas técnicas que podrían ayudarle a explorar con objetivo.

Personas, trabajos y roles

¡Probar un nuevo producto con nuevos ojos es un regalo! No hay tantas ideas preconcebidas sobre cómo debería comportarse el producto y el sesgo de confirmación de la gente no es tan fuerte. Un nuevo par de ojos puede observar mejor el producto objetivamente, aunque, por supuesto, todo el mundo tiene sesgos cognitivos inconscientes. Lisa ha notado que cuando se asocia con nuevos evaluadores de su equipo, inmediatamente notan errores que han estado ahí todo el tiempo, ¡pero nadie más podía “verlos”!

Asumiendo una persona , o un rol, permite a un miembro del equipo probar un producto que conoce por dentro y por fuera con una nueva perspectiva; Sesgos cognitivos inconscientes como la ceguera por falta de atención¹ y el sesgo de confirmación¹¹. se puede superar.

Una persona es un usuario ficticio que el equipo crea con características como edad, formación académica, experiencia, peculiaridades de personalidad, profesión, etc. Algunos equipos tienen un conjunto definido de personas que representan su base de clientes objetivo y que utilizan al diseñar nuevas funciones. La Figura 6.1 muestra un tipo de personaje de hacker que podría utilizar para realizar pruebas.

<https://www.stickyminds.com/article/how-pragmatic-personas-help-you-understand-your-end-user>

¹ http://www.theinvisiblegorilla.com/gorilla_experiment.html

¹¹https://en.wikipedia.org/wiki/Confirmation_bias



Figura 6.1: Persona hacker

Combinar personas con trabajos o roles es aún mejor para las pruebas exploratorias. He aquí un ejemplo.

Jill, una asistente ejecutiva, tiene 30 años, siempre tiene prisa y tiene mucho que hacer, y busca atajos mientras usa el producto. Pruebe la aplicación de reserva de hotel de su equipo como Jill, que está reservando un hotel en el último minuto para su jefe.

Cuando un evaluador asume la personalidad de Jill, es probable que utilice las capacidades de la función de una manera diferente a como lo haría normalmente. Por ejemplo, podría descubrir que al hacer clic en el botón Enviar varias veces por impaciencia se generan reservas duplicadas.

Flujos de trabajo y recorridos

Una forma común de explorar es recorrer los flujos de trabajo esperados o los recorridos de los usuarios en la aplicación. Comience con un viaje y luego explore sus variaciones. En esa aplicación de reserva de hotel, un viaje obvio es buscar una ubicación específica y un rango de fechas para una cierta cantidad de huéspedes, elegir una habitación e ingresar la información de la dirección para confirmar. Una variación de ese enfoque sería intentar ingresar una dirección de un país diferente. ¿El formulario acepta múltiples formatos de código postal? ¿Hace validación del código postal versus la dirección postal?

Otro enfoque popular es utilizar recorridos. Compárelo con hacer un recorrido por un destino de viaje. Como turista, si va a París, es posible que le guste ver varios lugares emblemáticos: la Torre Eiffel, el Louvre, el Arco de Triunfo o incluso la catedral de Notre Dame.

Una vez que hayas hecho eso, repite el recorrido pero ve a los lugares de interés en un orden diferente. ¡Las cosas pueden verse diferentes! Lo mismo sucede en el software. Probar el [recorrido histórico¹²](#) utiliza diferentes características y capacidades en diferentes órdenes y puede provocar un comportamiento inesperado.

Sugerencia: busque “tours de prueba exploratorios” en Internet y encontrará muchas ideas diferentes; le sugerimos que diseñe las suyas propias.



Figura 6.2: Recorrido por lugares emblemáticos

Riesgos y valor para el cliente.

Los equipos suelen pasar por alto los riesgos comerciales o lo que es de valor para un cliente. Se pueden diseñar sesiones de prueba exploratorias para centrarse en estos aspectos y descubrir suposiciones ocultas. Haciendo la pregunta: "¿Qué es lo peor que puede pasar?" puede exponer un riesgo que necesita ser explorado. Por ejemplo, si el robo de datos de los clientes supone un riesgo enorme, entonces el equipo quiere dedicar más tiempo a explorar los aspectos de seguridad del producto.

La otra cara del riesgo es el valor, así que pregúntese: "¿Qué es lo mejor que puede pasar?" y explorar en torno a ese valor comercial.

¹²https://blogs.msdn.microsoft.com/james_whittaker/2009/04/06/tour-of-the-month-the-Landmark-tour/

Explorar en parejas o grupos.

La exploración puede ocurrir en cualquier momento. Los programadores pueden explorar durante la codificación para acortar su ciclo de retroalimentación en caso de problemas visuales, o los evaluadores puede explorar la compatibilidad del navegador. Sin embargo, recomendamos emparejarse con alguien para aprovechar al máximo la experiencia (Figura 6.2).



Figura 6.3: Emparejamiento

Una forma de explorar a nivel de funciones es con grupos. uno de lisa Los equipos a menudo reunían a las personas para sesiones de prueba ad hoc o exploratorias para obtener mayor confianza en las características principales o riesgosas. Colaborando probar problemas de concurrencia es un gran ejemplo de esto. Más ojos puestos el problema significa mayores posibilidades de encontrar algo.

Al igual que la programación de mafias, [las pruebas de mafias¹³](#) se puede utilizar para pruebas exploratorias. Esto significa que hay un conductor (un rol que rota cada pocos minutos) con varias personas ayudando haciendo preguntas o sugerencias. Múltiples perspectivas pueden descubrir impactos en otras partes del sistema.

Cartas

En su libro ¡Exploralo! Elisabeth Hendrickson detalla cómo Utilice cartas para realizar pruebas exploratorias eficaces. Los charters te ayudan Organice la información que necesita para conocer su aplicación.

¹³<https://www.stickyminds.com/article/amplified-learning-mob-testing>

en sesiones de tiempo adecuadamente enfocadas. Estos funcionan bien en combinación con personas, trabajos y roles.

La plantilla de Elisabeth se ve así:

Explorar <objetivo>

Con <recursos>

Para descubrir <información de valor para alguien>

Considerar los recursos (o tipos de variaciones, como a Janet le gusta pensar en ellos) que se van a utilizar es una buena manera de comenzar a escribir una carta. Por ejemplo, es posible que sea necesario realizar pruebas de seguridad para probar varios exploits de formato. Se puede redactar una carta para explorar varias páginas de la interfaz de usuario con estos exploits. El siguiente ejemplo muestra una posibilidad.

Explora la página de registro de usuarios durante 30 minutos

Con exploits de secuencias de comandos entre sitios

Para descubrir cualquier vulnerabilidad

Nos gusta programar nuestras sesiones para ayudar a enfocar la carta.

Una forma sencilla de hacerlo es agregar el límite de tiempo al propio estatuto, como en nuestro ejemplo anterior.

A Lisa le gusta presentarles a las personas las pruebas exploratorias haciéndolas formar grupos pequeños para probar juguetes y juegos para niños pequeños.

Primero crean una personalidad, como por ejemplo: "Judy es una niña de cuatro años muy fuerte y activa". Luego prueban un juego diseñado para edades de 3 a 6 años con una carta:

Explora el juego como Judy.

Usando toda su energía, movimientos inesperados y fuerza.

Para descubrir si el juego es seguro para su grupo de edad.

Si pueden romper un trozo pequeño que representa un peligro de asfixia, es posible que el juego no sea seguro. Esa es una forma diferente de realizar la prueba que si simplemente usaras el juego como tu yo adulto.

Hay otras formas de redactar cartas. Algunos equipos usan mapas mentales, mientras que otros usan mnemónicos o simplemente escriben una oración sobre lo que quieren explorar.

Ejecutar, aprender, dirigir

Las personas a menudo se quedan atascadas al intentar redactar o ejecutar su primera carta.

Pista: Le sugerimos que si se encuentra en esta posición (atascado), no piense demasiado la primera vez. Escríbelo y luego pruébalo. Utilice sus habilidades de observación, su pensamiento crítico y su intuición.

A medida que se ejecutan los estatutos, el explorador aprende y puede escribir nuevos estatutos. También fomentamos la toma de notas. Una ventaja del emparejamiento es que la persona que no conduce puede escribir las notas. También creemos que reunirse con otros miembros del equipo después de la exploración es clave para aprender y compartir información.

Técnicas adicionales

Otras técnicas nos ayudan a “pensar fuera de lo común”. Por ejemplo, Mike Talks tiene sus tarjetas de “Prueba oblicua”¹ que puede ayudar al evaluador a recorrer un camino que de otro modo no se habría considerado.

TestSphere de Beren van Daele¹ Las tarjetas también hacen que los evaluadores piensen y hablen sobre sus pruebas de diferentes maneras. Como seres humanos, nuestros prejuicios inconscientes pueden impedirnos ver problemas importantes.

El uso de tarjetas como esta puede compensar esos prejuicios y ayudar a los equipos a ser más creativos.

¹ <https://leanpub.com/obliquetesting>

¹ <https://www.ministryoftesting.com/dojo/series/testsphere>

Aproveche las herramientas para una exploración eficaz

Las pruebas exploratorias están centradas en el ser humano, pero se pueden utilizar scripts o herramientas automatizadas para generar datos de prueba o preparar el escenario. Otras herramientas también pueden ayudar a explorar. Por ejemplo, los emuladores se pueden utilizar para dispositivos integrados o móviles, aunque también es necesario probar y explorar los dispositivos reales. Se pueden utilizar recursos como archivos de registro para detectar fallas "silenciosas" o posibles pérdidas de datos. Por ejemplo, uno de los equipos de Janet comenzó a resaltar las advertencias para hacerlas más visibles, lo que expuso un problema importante relacionado con el uso incorrecto de un método. Herramientas como las grabadoras pueden realizar un seguimiento de las páginas que se han visitado o de los datos utilizados, de modo que puedan reproducirse si se encuentra algo inesperado.

Capítulo 7: Pruebas

Atributos de calidad

Los atributos de calidad, o como a algunas personas les gusta llamarlos, requisitos no funcionales, a menudo se pasan por alto cuando se habla de una nueva característica o historia. Un atributo de calidad define las propiedades bajo las cuales debe operar una característica. En lugar de pensar en ellos como un “complemento”, preferimos pensar en ellos como una limitación que el equipo debe considerar en cada característica o historia.

Definición de atributos de calidad

Dos tipos principales de atributos de calidad que deben considerarse son los atributos operativos y de desarrollo. Los atributos de desarrollo incluyen la capacidad de mantenimiento, la reutilización y la capacidad de prueba del código: el “cómo” desarrollamos nuestro código. Se trata de calidad interna o tecnológica y es propiedad del equipo de entrega de software.

Cuando la gente habla de atributos de calidad, normalmente se refiere a los atributos operativos. Ellen Gottesdiener y Mary Gorman clasifican algunos de los atributos de calidad de la Figura 7.1 como atributos operativos o de desarrollo.

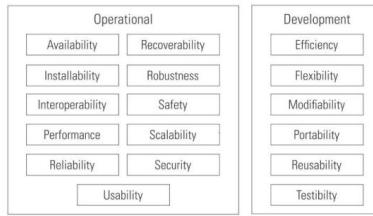


Figura 7.1: Metamodelo de atributos de calidad

Muchos de estos atributos de calidad están orientados a la tecnología (consulte el Capítulo 9: Cuadrantes de pruebas ágiles para obtener una explicación). Si las partes interesadas del negocio no los entienden bien, el equipo de entrega puede ayudarlos a establecer los niveles de calidad adecuados para cada atributo.

Mitigar los riesgos colaborando tempranamente

Cada producto u organización tiene necesidades y riesgos únicos que deben evaluarse. Al considerar qué atributos de calidad son importantes para sus clientes, el equipo puede hablar sobre los riesgos para el producto. Por ejemplo, Janet trabajó en un equipo donde la confiabilidad era el atributo de calidad más importante (aunque no el único). El equipo se preguntó: "¿Qué necesitamos para poder demostrar la fiabilidad?" Al responder la pregunta, la organización se dio cuenta de que necesitaban invertir en un entorno de pruebas de confiabilidad completo donde la compilación pudiera implementarse al final de cada iteración para ejecutar un conjunto completo de pruebas automatizadas junto con algunas pruebas exploratorias. El equipo trabajó para que las pruebas y simulaciones automatizadas funcionaran con cada historia.

Algunos equipos piensan en sus atributos de calidad después de entregar la funcionalidad y crean historias para los requisitos "no funcionales". Generalmente, esto no es una buena idea porque puede significar tener que volver atrás y rehacer la arquitectura o el diseño del código.

De hecho, estos atributos pueden tener mayor prioridad que los requisitos funcionales o de comportamiento. La valiosa funcionalidad en algunas áreas no supera la falta de seguridad o rendimiento en algunos dominios empresariales. Los equipos que esperan demasiado tarde en el ciclo para probar estos atributos (a menudo durante el final del juego, justo antes del lanzamiento) se topan con un obstáculo total.

Este tipo de problemas suelen ser problemas de diseño y no se pueden solucionar tan tarde en el ciclo de lanzamiento.

Como equipo de entrega, sea proactivo. No espere a que el propietario del producto inicie la conversación. Probablemente esté pensando en las capacidades de las funciones y dando por sentado los atributos de calidad. Como equipo, consideren qué aspectos de la calidad son más valiosos para los clientes y la empresa. Una buena manera de comenzar es dibujar un diagrama de contexto de la nueva característica propuesta para ver cómo interactúa con otras capacidades o sistemas. Conocer las dependencias y las áreas potencialmente frágiles con anticipación significa que hay tiempo suficiente para tomar las decisiones de diseño correctas y garantizar que el equipo tenga el conocimiento técnico necesario. El equipo podría planear hacer un pico (un experimento o una historia de investigación) para explorar diseños y arquitectura potenciales.

La planificación de lanzamientos o funciones ofrece grandes oportunidades para hacer preguntas a las partes interesadas del negocio como estas:

- ¿Qué es lo peor que puede pasar después de que liberemos esta capacidad? ¿Eso lo hace alto riesgo? • ¿Está bien si el sistema o una capacidad del sistema no funciona durante un período de tiempo? Si no, ¿cuál es el tiempo máximo o el porcentaje de tiempo que puede estar inactivo?

- Para una aplicación basada en web, ¿qué navegadores podrían usar los clientes? • ¿Podemos suponer que los clientes utilizarán dispositivos móviles?
 ¿Eso incluiría teléfonos y tabletas, y eso significa tanto Apple como Android? Qué pasa
- ¿Cómo sabremos si la función tiene éxito una vez que la lancemos?
 ¿él?

Planificación de las pruebas previas al lanzamiento

Es posible que algunos atributos de calidad requieran más pruebas inmediatamente antes del lanzamiento cuando todos los componentes del conjunto de funciones estén conectados. Por ejemplo, el equipo puede realizar pruebas de carga o rendimiento en un entorno de prueba para obtener una base final. Si el equipo usa [implementaciones azules/verdes¹](#) en una infraestructura de nube, pueden realizar estas pruebas en el entorno de producción inactivo.

Los equipos pueden utilizar alternancias de funciones de lanzamiento y otras técnicas para ocultar nuevas funciones a los clientes hasta que prueben varios atributos de calidad en producción. Una vez que estén seguros del nivel de calidad de todos los aspectos de la función, pueden activarla. (Vea más sobre pruebas en producción en el Capítulo 8: Pruebas en DevOps).

Planificación para el aprendizaje posterior

Haciendo la última pregunta: ¿Cómo sabremos si la función tiene éxito una vez que la lancemos? – es una excelente manera para que el equipo hable sobre el propósito de la nueva característica y cómo pueden medir si cumple con los objetivos deseados. El aprovisionamiento de datos para herramientas de análisis debe planificarse en las historias de cada función. A nivel de historia, piense en cómo el equipo puede monitorear el sistema para medir el uso y la calidad del atributo. Es posible que el equipo necesite nuevas herramientas para un registro y seguimiento adecuados.

A nivel de tarea, los programadores deberían pensar en instrumentar el código para que el equipo pueda medir con pruebas automatizadas o herramientas de monitoreo (ver Figura 7.2). La solución podría ser tan simple como ejecutar un optimizador en una nueva consulta de base de datos o usar una herramienta de análisis estático para verificar que el código cumpla con los estándares de accesibilidad. Un enfoque más holístico, como instrumentar cada evento.

¹ <https://docs.cloudfoundry.org/devguide/deploy-apps/blue-green.html>

en el código para que los problemas de producción puedan identificarse y solucionarse rápidamente, puede ser apropiado.

Ready	In Progress	To Review	Done
<div style="border: 1px solid black; padding: 2px;">Story 1</div> <div style="display: flex; justify-content: space-around; margin-top: 5px;"> <div style="border: 1px solid black; padding: 2px; width: 40px; height: 20px;"></div> <div style="border: 1px solid black; padding: 2px; width: 40px; height: 20px; background-color: #ffffcc;"></div> </div> <div style="display: flex; justify-content: space-around; margin-top: 5px;"> <div style="border: 1px solid black; padding: 2px; width: 40px; height: 20px;"></div> <div style="border: 1px solid black; padding: 2px; width: 40px; height: 20px;"></div> </div> <div style="display: flex; justify-content: space-around; margin-top: 5px;"> <div style="border: 1px solid black; padding: 2px; width: 40px; height: 20px;"></div> <div style="border: 1px solid black; padding: 2px; width: 40px; height: 20px;"></div> </div>			
<div style="border: 1px solid black; padding: 2px;">Story 2</div>			

Figura 7.2: Panel de tareas con el futuro en mente

Cumplimiento normativo

El cumplimiento normativo no siempre se considera un atributo de calidad, pero al igual que los atributos de calidad que hemos mencionado, es necesario considerarlo desde el principio. El cumplimiento no significa necesariamente montones de documentos, pero generalmente implica trabajo adicional para el equipo y es aconsejable planificar con anticipación.

Las organizaciones deben trabajar junto con auditores y agencias reguladoras para comprender qué información se requiere para demostrar el cumplimiento. Es importante que todos tengan la misma visión de un enfoque adecuadamente disciplinado. Por ejemplo, si el equipo tiene pruebas automatizadas que se ejecutan todos los días y las pruebas proporcionan documentación viva en forma de resultados de las pruebas, ¿pueden usarse como evidencia para respaldar el enfoque o la cobertura? Ambos hemos trabajado con equipos que necesitaban demostrar cumplimiento (dispositivos médicos y finanzas) y lo hicimos con muy poco trabajo "extra". Consulte el Capítulo 21, "Pruebas ágiles en entornos regulados" en Pruebas más ágiles para obtener más ejemplos e información.

Capítulo 8: Pruebas en DevOps

El desarrollo de software siempre ha incluido el proceso de poner en producción nuevos cambios en el software para que los utilicen los clientes.

En el pasado, gran parte de este proceso era manual. Existen nuevas herramientas y prácticas para crear artefactos de software e implementarlos en entornos de prueba y producción. Pero el proceso básico es el mismo. Los equipos realizan muchas actividades de prueba para ayudarlos a sentirse seguros acerca de los cambios que realizan en su producto de producción.

Hay términos nuevos para algunas de estas pruebas, pero las habilidades básicas de prueba siguen siendo relevantes.

El movimiento DevOps surgió de la idea de que algunas organizaciones habían adoptado el desarrollo ágil pero habían dejado a todo su personal de operaciones fuera de la transición. También es el resultado del cambio a aplicaciones e infraestructura alojadas en la nube a medida que el código reemplazó las interfaces de línea de comandos. Los roles se han adaptado. Los especialistas en operaciones aprenden a codificar. Los programadores asumen la responsabilidad de su código incluso después de que esté en producción, en lugar de arrojarlo por encima de la pared para las operaciones.

Los evaluadores también adaptan sus propias habilidades y actividades. Contribuyen de muchas maneras, como ayudar a diseñar conjuntos de pruebas automatizadas que brindan información confiable y valiosa, optimizar el proceso de entrega y probar el código de infraestructura para garantizar la confiabilidad de ejecución.

El capítulo 23 de Pruebas más ágiles detalla cómo los especialistas en operaciones pueden ayudar al equipo de entrega a mejorar la calidad mediante la configuración de entornos de prueba, ayudando a implementar marcos de automatización de pruebas, generando datos de prueba y más.

Entrega e implementación continuas

Los equipos que practican la entrega continua (CD) tienen una versión candidata implementable cada vez que se confirma un nuevo cambio en el repositorio de código y posteriormente pasa con éxito a través de un proceso de implementación. El proceso comienza con una integración continua, que puede incluir conjuntos de pruebas automatizadas en diferentes niveles, como unidad, API y flujo de trabajo completo a través de la interfaz de usuario.

Puede incluir otros pasos, como el análisis de código estático para implementaciones automatizadas en diversos entornos. Las partes interesadas del negocio pueden decidir implementar la versión candidata en producción y pueden hacerlo muchas veces al día. La Figura 8.1 muestra un ejemplo de un canal de entrega continua.

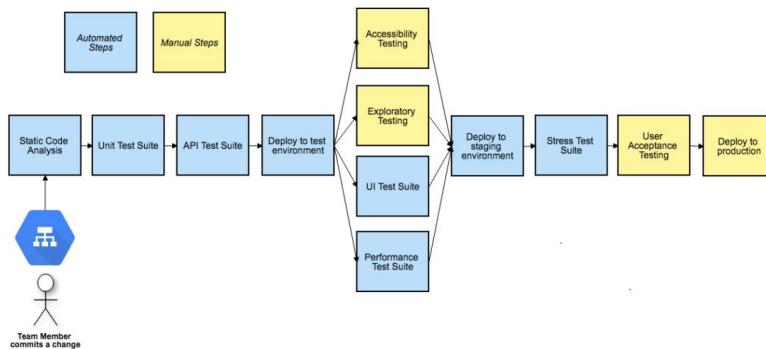


Figura 8.1: Canal de entrega continua

La implementación continua (también CD) es el mismo proceso, excepto que cada candidato de lanzamiento exitoso se implementa automáticamente en producción. La Figura 8.2 muestra un ejemplo de un proceso de implementación continua.

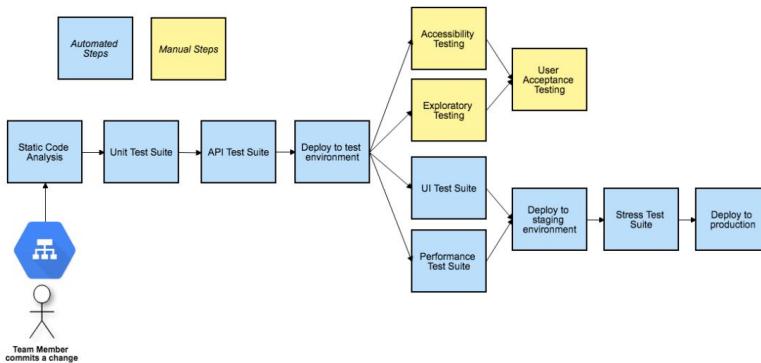


Figura 8.2: Canal de implementación continua

Esto suena tan aterrador como probar en producción si no lo has hecho antes. Si publica varias veces al día, ¿cómo puede realizar todas las pruebas que necesita realizar? Las pruebas centradas en el ser humano, ya sean pruebas que el equipo aún no ha automatizado o actividades de prueba como pruebas exploratorias y de accesibilidad, son una parte tan importante del proceso de entrega como las pruebas automatizadas.

La clave es reconocer la diferencia entre implementar y lanzar.

Gracias a técnicas como la alternancia de funciones de lanzamiento, es posible ocultar los cambios a los clientes hasta que se completen todas las pruebas necesarias. Las pruebas se pueden realizar de forma asincrónica.

La entrega o implementación continua representa un alto nivel de logro para un equipo. Los miembros del equipo de entrega y las partes interesadas del negocio deben crear una comprensión compartida de cada característica e historia que se entregará. Los desarrolladores dominan las formas de ocultar funciones a algunos (o todos) los clientes hasta que estén listos para lanzar esa función. El proceso de entrega debe ser rápido para poder implementarse diariamente o varias veces al día. Eso requiere una buena infraestructura, lo que significa más código para construir y probar.

Hay muchas habilidades nuevas que dominar. Cuando cada miembro del equipo

trae sus [habilidades en forma de T¹](#) y todos colaboran, el equipo puede resolver problemas más fácilmente y continuar acortando los ciclos de retroalimentación representados en el proceso.

Pruebas en producción

Hubo un tiempo en que el término "prueba en producción" sonaba como "Lanzar el código a producción y dejar que los clientes encuentren los errores y nos lo digan" o "Probemos en producción y esperemos que no afectemos la cuenta de nadie".

Lamentablemente, algunos equipos hicieron precisamente eso. Hoy en día, las palabras "pruebas en producción" tienen una connotación menos aterradora.

Las pruebas en producción se han convertido en una necesidad en muchos casos, ¡pero eso no significa publicar código de mala calidad y dejar que los clientes encuentren los errores!

Las pruebas en producción ayudan a las empresas de múltiples maneras. Generalmente es imposible crear un entorno de prueba que se parezca exactamente al de producción. No existe una manera fácil de saber realmente qué hará el software hasta que esté en el entorno de producción.

Técnicas como la alternancia de funciones de lanzamiento permiten a los equipos "activar" funciones específicas para clientes específicos para obtener comentarios rápidos. A veces esto se denomina versión de aprendizaje o producto mínimo viable (MVP). Las pruebas A/B pueden ser la técnica de prueba en producción más conocida, mostrando diferentes diseños a diferentes personas y juzgando cuál genera la mayor cantidad de "clics" o ventas.

Los análisis y el seguimiento sofisticados pueden mostrar detalles como cómo un usuario individual navega a través de la aplicación o estadísticas agregadas como qué porcentaje de clientes están utilizando una capacidad específica. Eliminar funciones no utilizadas puede ser tan importante como agregar otras nuevas y populares, ya que cada línea de código tiene un costo de mantenimiento y agrega riesgos. Las pruebas en producción consisten en monitorear y observar.

¹ <https://lisacrispin.com/tag/t-shed-skills/>

Monitoreo y observabilidad

La importancia de monitorear el estado del sistema en producción ha existido desde que los sistemas de software han registrado información pertinente sobre eventos en el sistema. Los equipos pueden configurar alertas para ciertos tipos de errores o para exceder un presupuesto de errores; por ejemplo, "Publicar una alerta cuando la cantidad de errores 503 aumente un 10 % con respecto al promedio". Cuando se les alerta, los miembros del equipo profundizan en los archivos de registro y análisis para investigar el problema y poder resolverlo.

Los probadores profesionales saben que no es posible predecir todos los posibles errores que pueden ocurrir en la producción! En los últimos años, esto ha dado lugar a una nueva práctica llamada observabilidad, a menudo denominada "o11y", que representa los 11 caracteres entre la 'o' y la 'y' en el alfabeto inglés. Los equipos que practican o11 instrumentan y registran cada evento en su código de producción para que pueda ser analizado por las herramientas adecuadas cuando sea necesario. Utilizando herramientas y experimentos, los equipos estudian información de datos de registro estructurados, métricas y seguimientos para aprender cosas diferentes sobre su producto de las que se pueden aprender en un entorno de prueba. Esta es un área donde los evaluadores, con su capacidad para detectar patrones inusuales e identificar riesgos, aportan valor. Como Abby Bangser¹ Como dijo, es una forma de prueba exploratoria asistida por herramientas.

La observabilidad permite a los equipos responder rápidamente cuando un usuario informa un problema que el equipo nunca antes había visto o cuando las métricas del sistema muestran patrones inusuales que indican un problema potencial. Los datos de registro estructurados permiten al equipo rastrear la actividad del usuario e identificar rápidamente dónde el sistema comenzó a comportarse incorrectamente. Esto permite a los equipos revertir un cambio o implementar una solución, a menudo en cuestión de minutos. Esta capacidad de recuperarse tan rápidamente de fallas de producción permite al equipo implementar estos pequeños cambios continuos en el producto sin miedo, y es lo que hace que la entrega o implementación continua sea una práctica segura.

¹ <https://club.ministryoftesting.com/t/power-hour-curious-stuck-or-need-guidance-on-devops-or-observability/24963/3>

La nueva tecnología nos trae nuevas capacidades

En los últimos años, el almacenamiento de “grandes datos” se ha vuelto asequible incluso para las pequeñas empresas. Los equipos pueden registrar información sobre cada evento que ocurre en cualquier entorno de prueba o producción. La poderosa tecnología, incluida la inteligencia artificial (IA) y el aprendizaje automático (ML), ha acelerado el procesamiento y análisis de estas enormes cantidades de datos. Podemos aprender rápidamente cómo los clientes utilizan nuestros productos, descubrir problemas antes que ellos y recuperarnos rápidamente de las fallas.

Las pruebas en producción son solo una parte del enfoque de cualquier equipo para incorporar calidad a su producto. Todavía planifican y ejecutan todas las actividades de prueba apropiadas junto con la escritura del código de producción. También pueden observar el uso en producción y ejecutar pruebas sin riesgos en producción para agregar otro nivel de confianza y confiabilidad.

La Figura 8.3 es un gráfico maravilloso de Cindy Sridharan de su artículo [Monitoreo y Observabilidad](#)¹, y representa cómo los equipos prueban tratando de simular la producción, monitorean fallas predecibles en la producción y usan la observabilidad para detectar cualquier cosa que se escape de esos otros esfuerzos impulsados por la calidad.

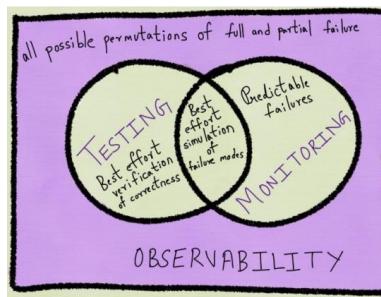


Figura 8.3: Pruebas, seguimiento y observabilidad de Cindy Sridharan

¹ <https://medium.com/@copyconstruct/testing-in-production-the-safe-way-18ca102d0ef1>

SECCIÓN 3: Útil Modelos

Hemos descubierto que los modelos visuales son un ingrediente esencial para ayudar a los equipos a planificar y ejecutar todas las actividades de prueba necesarias y formular una estrategia de automatización de pruebas eficaz. En esta sección, explicamos cómo utilizar los cuadrantes de pruebas ágiles para identificar todos los tipos de pruebas que se necesitan para cada nuevo conjunto de características o historia, y asegurarnos de que se realicen cuando sean más efectivas. Luego, analizamos cómo utilizar modelos visuales como la Pirámide de automatización de pruebas para guiar las conversaciones del equipo sobre una estrategia de automatización realista que funcione para su contexto.

- Capítulo 9: Los cuadrantes de pruebas ágiles •
Capítulo 10: Visualización de una estrategia de automatización de pruebas

Capítulo 9: El ágil Cuadrantes de prueba

Brian Marick escribió por primera vez sobre los cuadrantes de pruebas ágiles (Figura 9.1) en 2003, cuando Lisa y Janet intentaban descubrir cómo hablar sobre las pruebas a la comunidad ágil, especialmente a los evaluadores. En ese momento, la mayoría de la gente ágil hablaba sólo de “pruebas de clientes” y “pruebas de programadores”. Muchos equipos se centraron únicamente en las pruebas de aceptación funcional: cómo deberían comportarse las funciones.

Hoy en día, esto sigue siendo un peligro potencial para los equipos que hacen la transición a la metodología ágil. Los cuadrantes nos brindaron una manera de discutir los diferentes tipos de pruebas que un equipo podría necesitar considerar. Nos ayudan a ver el panorama general.

Los cuadrantes de pruebas ágiles son una taxonomía de diferentes tipos de pruebas. Lo utilizamos como una herramienta de pensamiento para ayudar a los equipos a discutir qué actividades de prueba podrían necesitar y asegurarnos de que tengan las personas, los recursos y los entornos adecuados para realizarlas.

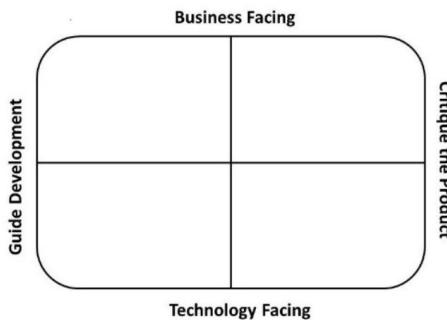


Figura 9.1: Cuadrantes de pruebas ágiles

Las pruebas del lado izquierdo son aquellas que guían el desarrollo, las que se escriben antes de que se produzca la codificación o al mismo tiempo que avanza la codificación. Las pruebas del lado derecho son aquellas que critican (evalúan) el producto una vez completada la codificación. Las pruebas de la izquierda ayudan a prevenir defectos. Las pruebas de la derecha encuentran defectos en el código o quizás identifican características faltantes.

La mitad superior de los cuadrantes se centra en pruebas que las partes interesadas del negocio pueden leer. Estas pruebas responden a la pregunta: "¿Estamos construyendo lo correcto?" La mitad inferior incluye pruebas escritas por y para miembros del equipo técnico. Las partes interesadas del negocio probablemente se preocupan por los resultados finales, pero no intentarían leer las pruebas. La mitad superior trata sobre la calidad externa definida por la empresa. La mitad inferior trata sobre el código interno o la corrección de la infraestructura.

Los cuadrantes están numerados para facilitar la referencia. Los cuatro cuadrantes están etiquetados como:

- Cuadrante 1 (Q1): Pruebas orientadas a la tecnología que guían el desarrollo.
opción
- Cuadrante 2 (Q2): Pruebas orientadas al negocio que guían el desarrollo.
mento
- Cuadrante 3 (Q3): Pruebas orientadas al negocio que critican el producto.
producto
- Cuadrante 4 (Q4): Pruebas orientadas a la tecnología que critican la
producto

El modelo ágil de cuadrantes de prueba ayuda a los equipos a pensar en las actividades de prueba necesarias para dar confianza al producto que están creando. También ayuda a crear un lenguaje de prueba común con el equipo y con la organización si se utiliza para ayudar a comunicarse entre equipos. Lo que más le gusta a Janet de este modelo es que no solo representa una visión holística de las pruebas, sino que también hace visible la responsabilidad de todo el equipo por las actividades de prueba.

Cada equipo tiene su propia combinación distintiva de dominio comercial, producto, habilidades, madurez del producto, conjunto técnico, supervisión regulatoria y más. El modelo se puede aplicar para representar las pruebas requeridas en cada contexto. En la Figura 9.2, compartimos algunos tipos típicos de pruebas que pueden encontrarse en cada cuadrante.

Sugerencia: Recuerde: utilice estos ejemplos como guía. Es una herramienta, no una regla, y hay muchas zonas grises. El contexto de su equipo es único y sus cuadrantes también deberían serlo. Aplique el modelo a las pruebas que necesita realizar.

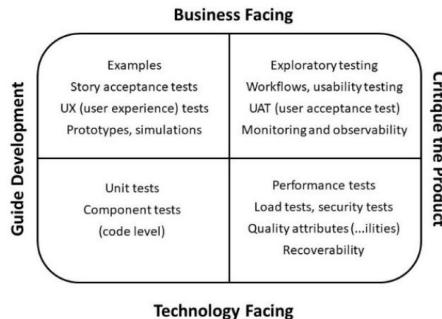


Figura 9.2: Ejemplos de tipos de pruebas para los cuadrantes

¿Qué pruebas y en qué orden?

Hemos numerado los cuadrantes 1, 2, 3, 4 simplemente para facilitar la referencia. Es complicado decir "Pruebas empresariales que guían el desarrollo", por eso decimos "Cuadrante 2" o "Q2". Los números no pretenden representar que los tipos de actividades de prueba deben realizarse en ese orden. A medida que el equipo planifique sus pruebas, pensarán en el momento apropiado para realizar cada actividad de prueba.

He aquí un ejemplo. Un equipo decide comenzar a utilizar una nueva arquitectura para las nuevas funciones en el futuro. Necesitan estar seguros de que

la arquitectura se escalará adecuadamente para acomodar una cierta cantidad de usuarios y una carga potencial en el sistema. El equipo hace un "pico", lo que significa que escriben código desechable únicamente con el fin de probar la arquitectura. Realizan pruebas de carga y rendimiento en el "pico" para ver si cumple con los requisitos de tiempo de respuesta y se mantiene estable. Si están satisfechos, eliminan el pico y comienzan a trabajar en historias para una nueva función, esta vez siguiendo sus prácticas de desarrollo habituales. Si no están satisfechos, pueden repensar la arquitectura y repetir el proceso.

Para el desarrollo de funciones, la mayoría de los equipos probablemente comiencen en el segundo trimestre. Podrían dibujar prototipos en papel y mostrárselos a clientes potenciales. El equipo de entrega y negocios podría tener un taller de mapeo de historias o especificaciones para discutir las características y dividirlas en historias. Durante los talleres de refinamiento del trabajo pendiente o de preparación de historias, el equipo puede utilizar actividades como el mapeo de ejemplos para extraer más detalles.

Una vez que el equipo comienza a desarrollar una historia, trabajan en las pruebas desde el primer trimestre y escriben pruebas unitarias como parte del desarrollo basado en pruebas. Es posible que estén trabajando simultáneamente en pruebas de historias en el segundo trimestre. Si se entrega una característica suficiente para explorar, es posible que pasen a las pruebas del tercer trimestre. Sin embargo, si la seguridad es la principal preocupación para la empresa y los clientes, las pruebas de seguridad en el cuarto trimestre pueden tener prioridad sobre las pruebas de historias funcionales del segundo trimestre. Cada equipo encuentra su propia forma de trabajar y puede cambiar según diferentes tipos de funciones o proyectos.

Usando los cuadrantes

Janet y Lisa han visto equipos usar los cuadrantes de muchas maneras.

Algunos equipos colocan un cartel en la pared con los cuadrantes en blanco. Mientras planifican una nueva característica o lanzamiento, hablan sobre todas las pruebas que necesitarán y escriben cada tipo en el cuadrante apropiado. Lo utilizan como recordatorio de lo que deben hacer o de lo que pueden haber olvidado.

Cuadrante 1

Los equipos también pueden utilizar los cuadrantes para hablar sobre qué pruebas deberían automatizarse. Las pruebas del primer trimestre suelen ser automatizadas por los desarrolladores que escriben el código de producción. Muchos equipos practican el desarrollo basado en pruebas (TDD), escribiendo una pequeña prueba unitaria para alguna pequeña pieza de funcionalidad y luego el código para hacer que esa prueba pase. Las pruebas del primer trimestre están diseñadas para ejecutarse rápidamente, ya que prueban un área de código muy pequeña y generalmente no incluyen interacción con otras capas de la aplicación o bases de datos. Brindan a los equipos la información rápida necesaria para realizar cambios en el código de forma rápida y sin miedo.

Cuadrante 2

Las pruebas comerciales que guían el desarrollo en el segundo trimestre son una base importante para la mayoría de los equipos. Los propietarios de productos, desarrolladores, evaluadores y otros se reúnen con frecuencia para planificar funciones e historias. Pueden utilizar técnicas como el mapeo de ejemplos para obtener reglas de negocio para cada historia junto con los ejemplos que las ilustran. Los equipos que practican el desarrollo basado en el comportamiento (BDD), el desarrollo basado en pruebas de aceptación (ATDD) o la especificación por ejemplo (SBE) los convierten en escenarios que especifican el comportamiento como pruebas ejecutables. Estos escenarios se pueden automatizar a medida que el código es escrito.

Cuadrante 3

Las pruebas en el tercer trimestre tienden a centrarse en las personas, y determinan si las historias y características entregadas brindan el valor deseado a los clientes. Se puede utilizar la automatización para facilitar estas pruebas mediante la configuración de datos o estado. Cada vez es más común que los equipos realicen pruebas exploratorias en producción, utilizando funciones de lanzamiento para "ocultar" nuevas funciones a los clientes hasta que se completen las pruebas. Las pruebas del tercer trimestre incluyen formas de pruebas en producción, como análisis de seguimiento para

aprenda qué sucede realmente y cómo los clientes usan las funciones.

La información aprendida en las pruebas del tercer trimestre se retroalimenta en el segundo, lo que a menudo resulta en la creación de nuevas historias o características.

Cuadrante 4

Muchas pruebas del cuarto trimestre dependen de la automatización y las herramientas, pero algunas pueden requerir pruebas adicionales. Por ejemplo, las herramientas automatizadas para las pruebas de accesibilidad (a menudo abreviadas como "a11y", que representan las letras inicial y final y el número de letras intermedias) aún no son tan efectivas como las pruebas exploratorias manuales para esas capacidades. Los resultados de estas pruebas a menudo se retroalimentan en las actividades del primer trimestre a medida que el equipo cambia el diseño del código para mejorar varios atributos de calidad. La supervisión del rendimiento y los errores en la producción, o las pruebas de recuperabilidad, también pueden considerarse un tipo de prueba que se incluye en el cuarto trimestre. La tecnología actual ha hecho posible y seguro realizar pruebas en producción. Esto no significa que dejemos que los clientes encuentren errores por nosotros, sino que aprendemos con seguridad cómo se comporta el código en un entorno de producción real. (Los detalles sobre las pruebas en producción se pueden encontrar en el Capítulo 8).

Definición de "Listo"

Utilice los cuadrantes para definir qué significa "Listo" para su equipo.

Muchos equipos tienen dificultades al intentar decidir qué pruebas deberían incluirse en esa definición. En el Capítulo 3 hablamos de niveles de detalle, y esos niveles se aplican aquí. En lugar de definir "Terminado", animamos a los equipos a ser más específicos y llamarlo "Historia terminada".

Las pruebas que se pueden incluir en "Historia terminada" probablemente serían todas las del primer trimestre, todas las del segundo trimestre y quizás algunas del tercer trimestre, como pruebas exploratorias.

Vaya un paso más allá y defina "Función terminada", que incluiría todas las pruebas de las historias, pero quizás también incluya pruebas como pruebas de aceptación del usuario (UAT) y pruebas exploratorias en la función.

nivel. Recomendamos que todos los atributos de calidad que no se pudieron probar a nivel de historia se realicen a nivel de característica.

Incluso puede optar por definir "Liberación realizada". Eso incluiría todas las pruebas que se apliquen en su contexto, de cada cuadrante.

Sugerencia: recuerde, cuando hablamos de "finalización" en los niveles de historia, función y lanzamiento, no estamos exigiendo que las pruebas se realicen en un orden determinado. Más bien, considere qué pruebas guían el desarrollo (previenen defectos) y cuáles critican el producto (encontrar defectos en el código).



Figura 9.3: Encontrar y "corregir" errores

Encuentra los modelos que se adaptan a tu contexto

Muchos equipos han encontrado útiles los cuadrantes de pruebas ágiles para planificar sus actividades de prueba. Otros han adaptado el modelo de cuadrantes para adaptarlo mejor a sus necesidades y existen muchas variaciones útiles en los cuadrantes. Puede encontrar el Capítulo 8 de More Agile Testing disponible para descargar en agiletester.ca² . e incluye varias adaptaciones de los cuadrantes. Consulte nuestra lista de recursos para más.

Cualquiera que sea el modelo que funcione mejor para usted, asegúrese de mantenerlo visible y utilizarlo para estimular conversaciones sobre cómo mejorar continuamente sus pruebas.

² <https://agiletester.ca>

Capítulo 10: Visualizando un Automatización de pruebas Estrategia

La automatización de pruebas es un desafío constante para los equipos de software en todas partes. Muchos equipos aún no cuentan con pruebas de regresión automatizadas. Algunos equipos sienten que dominan el proceso, pero luego actualizan su producto para incorporar nueva tecnología que sus herramientas de automatización existentes no pueden manejar. A menudo tienen dificultades para mantener un equilibrio entre el valor y el coste de mantenimiento.

Dondequiera que un equipo individual se encuentre en su viaje de automatización, es útil dar un paso atrás y pensar en las prioridades y en qué mejoras centrarse a continuación. Este capítulo no pretende brindarle una introducción extensa a la automatización, sino mostrar cómo el uso de modelos visuales puede ayudar a los equipos a diseñar su estrategia de automatización.

Usando modelos visuales

Involucrar a todo el equipo en la formulación de una estrategia para abordar las diferentes necesidades de automatización y ejecutar esa estrategia es clave para tener éxito con la automatización. Los modelos visuales ayudan a guiar estas conversaciones.

El modelo de cuadrantes de pruebas ágiles que se trata en el Capítulo 9 puede ayudar a los equipos a planificar su estrategia de automatización mientras analizan los diferentes tipos de pruebas que se necesitan, así como qué habilidades, herramientas e infraestructura necesitarán para completarlas. En este capítulo, miramos

en algunos modelos adicionales que pueden ayudar a los equipos a encontrar una estrategia de automatización exitosa.

Sugerencia: recuerde, estas son herramientas de pensamiento que se pueden utilizar para iniciar conversaciones sobre cómo quiere trabajar su equipo. automatizar pruebas.

La clásica pirámide de automatización de pruebas

La pirámide de automatización de pruebas de Mike Cohn ha ayudado a muchos equipos desde principios de la década de 2000. Lo hemos ajustado ligeramente desde entonces (Figura 10.1) para dejar clara nuestra intención, incluyendo la burbuja de la nube en la parte superior para representar que no todas las pruebas de regresión se pueden automatizar. A veces necesitamos pruebas centradas en el ser humano, que incluyen pruebas exploratorias (ET).

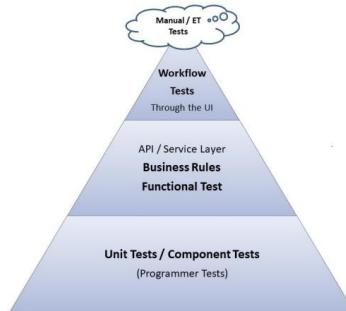


Figura 10.1: Pirámide de automatización de pruebas clásica

Este modelo ayuda a los equipos a comprender que, en la mayoría de los contextos, vale la pena automatizar las pruebas en el nivel más granular posible de la aplicación, para brindar una protección adecuada contra fallas de regresión.

Los equipos que practican el desarrollo basado en pruebas (TDD) crean una base sólida de pruebas a nivel de unidad y componente que ayudan a guiar el diseño del código.

Estas pruebas se ejecutan muy rápido, por lo que brindan al equipo información rápida.

Con la mayoría de las aplicaciones, es necesario probar las interacciones entre diferentes capas de la arquitectura. Por ejemplo, la lógica empresarial.

normalmente requiere interacción con la base de datos. Realizar la mayor cantidad de automatización de este tipo a nivel de servicio o API sin pasar por una interfaz de usuario (UI) es generalmente la forma más eficiente.

Algunas regresiones sólo ocurren cuando están involucradas dos o más capas de la aplicación. Esto puede requerir pruebas de flujo de trabajo a través de la interfaz de usuario que involucren al servidor, la base de datos y/o un sistema externo.

En la mayoría de los contextos, es mejor minimizar la cantidad de pruebas de flujo de trabajo de un extremo a otro. Estas pruebas se ejecutan lentamente, suelen ser las más frágiles y, por lo general, requieren la mayor cantidad de mantenimiento. La nube en la cima de la pirámide incluye actividades centradas en el ser humano, como pruebas exploratorias y otras tareas que no se pueden automatizar.

La pirámide de automatización de pruebas nos ayuda a pensar en formas de "reducir las pruebas", maximizar las pruebas de regresión que están aisladas en una parte de una aplicación y minimizar aquellas que involucran varias partes del sistema. Cuando los equipos planean probar una función, pueden observar la pirámide para ver dónde se puede automatizar más apropiadamente cada prueba.

La pirámide clásica no pretende implicar que exista un cierto número o porcentaje de pruebas automatizadas en cada nivel. Seb Rose visualiza el modelo de manera un poco diferente, como se muestra en la Figura 10.2.

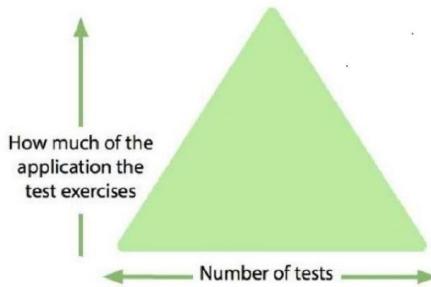


Figura 10.2: Versión de la pirámide de Seb Rose, número de pruebas versus cobertura de pruebas

El modelo de Seb aclara que lo que encarece una prueba es el número de capas de la aplicación que requiere para ejecutarse. Para

Por ejemplo, es posible realizar una prueba a nivel unitario de la interfaz de usuario que no involucre ninguna otra capa de la aplicación. Es posible utilizar TDD para cada capa aislada de la aplicación, ya sea el servidor, la API, la UI o un microservicio.

Ha habido muchas adaptaciones del modelo piramidal (y sí, el clásico es en realidad un triángulo) a lo largo de los años. El capítulo 15 de More Agile Testing incluye varias adaptaciones, incluidas las de Alister Scott y Sharon Robson.

Los equipos que tienen pruebas automatizadas pueden dibujar la "forma" de su pirámide para visualizar dónde encajan sus pruebas actuales. Muchos equipos comienzan principalmente con pruebas de interfaz de usuario y una pirámide "invertida" o "[cono de helado²¹](#)". Otros pueden tener un reloj de arena. Ninguna de estas formas es necesariamente incorrecta, pero si la automatización actual no satisface las necesidades del equipo, las imágenes pueden ayudar a imaginar qué cambios se necesitan.

Las conversaciones sobre un modelo visual ayudan a los equipos que recién están comenzando sus esfuerzos de automatización a decidir su objetivo final y sus primeras prioridades.

Pruebas automatizadas como documentación viva

El tiempo, el costo y el esfuerzo que se dedican a automatizar diferentes tipos de pruebas en diferentes niveles de la aplicación no son las únicas consideraciones al elaborar una estrategia de automatización.

Otra consideración es recordar quién debe poder leer y comprender las pruebas. El Iceberg de automatización de pruebas de Seb Rose (Figura 10.3) nos recuerda que uno de los atributos más valiosos de las pruebas automatizadas es la documentación viva que proporcionan.

Siempre están actualizados porque el equipo los mantiene pasando todos.

²¹<https://watirmelon.blog/testing-pyramids/>

la hora, para que pueda saber en cualquier momento exactamente qué hace su sistema.

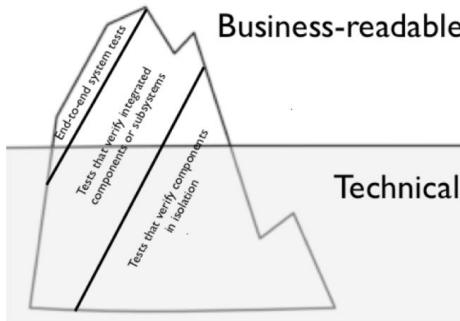


Figura 10.3: Iceberg de automatización de pruebas de Seb Rose

Las partes del iceberg que se encuentran por encima de la línea de flotación son pruebas legibles para los negocios, mientras que las que se encuentran debajo no lo son (están escritas en un lenguaje técnico). La cantidad de pruebas variará según el equipo; por ejemplo, Lisa trabajó en un equipo cuyo producto estaba destinado a otros equipos de entrega. Todos los miembros del equipo, incluido el propietario del producto, podían comprender las pruebas de regresión automatizadas escritas en código de bajo nivel. No necesitaban pruebas “legibles para los negocios”. En otros ámbitos, es fundamental que las partes interesadas del negocio puedan comprender las pruebas de aceptación. Este modelo nos ayuda a recordarnos que debemos pensar en lo que se necesita en el contexto del equipo.

Ampliando el modelo

Incluso la cobertura de pruebas de regresión automatizada más diligente puede no identificar algunas fallas de regresión. Ningún entorno de prueba es exactamente como el de producción. Algunos errores se pueden encontrar mediante “pruebas en producción”, como se analiza en el Capítulo 8.

En su libro Una guía práctica para realizar pruebas en DevOps, Katrina Clokie presenta su filtro de errores de DevOps. Es una imagen útil que muestra que las pruebas unitarias solo pueden filtrar los errores pequeños, mientras que los diferentes niveles de

Las pruebas de integración y de extremo a extremo detectan los cada vez más grandes. Para encontrar los errores completamente formados, los equipos necesitan registros, alertas y monitoreo para su sistema de producción.

Responsabilidad compartida

Alentamos a los equipos a compartir la responsabilidad de probar las reglas comerciales y niveles más altos de integración a nivel de API.

Los evaluadores también deben tener visibilidad de las pruebas unitarias y de componentes escritas por los desarrolladores. Como miembro del equipo, es importante comprender las aplicaciones de su equipo y el funcionamiento interno al abordar su estrategia de automatización.

Debido a que la automatización de las pruebas a través de la interfaz de usuario tiende a llevar más tiempo, existe la tentación de delegar eso a un equipo de automatización separado o hacer que los evaluadores del equipo asuman toda la responsabilidad.

Recomendamos que los desarrolladores, que son buenos escribiendo código eficiente y fácil de mantener, trabajen junto con los evaluadores, que son buenos especificando casos de prueba, para automatizar las pruebas a través de la interfaz de usuario, así como todas las demás capas por encima del nivel base de la pirámide.

Recuerde, como todos los demás modelos, la pirámide de automatización de pruebas es una guía. Los equipos que reúnen a miembros de todos los roles para hacer preguntas, conversar sobre las respuestas, dibujar en la pizarra y diseñar experimentos tienen mayor éxito con la automatización. Los modelos visuales como los ejemplos de este capítulo ayudan a los equipos a hablar sobre por qué están automatizando pruebas, cuáles son sus mayores puntos débiles de la automatización, cómo podrían ayudar las personas con diferentes habilidades y cuáles deberían ser sus próximos experimentos. Según nuestra experiencia, un enfoque paso a paso funciona mejor.

SECCIÓN 4: Pruebas ágiles Hoy

Los conceptos básicos de las pruebas ágiles, como utilizar el enfoque de todo el equipo, guiar el desarrollo con ejemplos y colaborar entre roles para generar calidad, son tan efectivos hoy como lo eran hace 20 años.

¿Hay algo que debamos cambiar para enfrentar los desafíos actuales? En esta sección, compartimos lo que algunos profesionales líderes en pruebas ágiles ven como cambios en el rol de los evaluadores. Terminaremos con una serie de ingredientes para ayudar a los equipos a tener éxito con las pruebas ágiles.

- Capítulo 11: El nuevo rol de un evaluador
- Capítulo 12: Ingredientes para el éxito

Capítulo 11: Un probador Nuevo rol

Muchos equipos luchan con un solo tester como parte del equipo o, peor aún, con un tester que apoya a más de un equipo de entrega.

Si el evaluador es el único que realiza actividades de prueba, generalmente crea un cuello de botella. Los equipos ágiles que realizan cambios en la producción en cada iteración (o incluso con mayor frecuencia si implementan la entrega continua) no pueden permitirse el lujo de tener un solo evaluador para realizar todas las pruebas.

No podemos predecir el futuro, pero es informativo mirar a nuestro alrededor para ver cómo está cambiando el papel del evaluador. Le pedimos a otros entrenadores y capacitadores con experiencia en pruebas ágiles que descubrieran qué piensan sobre el papel cambiante de un evaluador. Estas diferentes perspectivas pueden ayudarle a comprender cómo los evaluadores y los equipos pueden adaptarse y ayudar a construir una cultura de calidad.

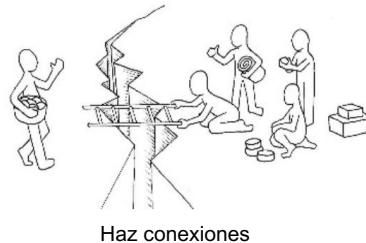
Los probadores son pegamento de calidad para un equipo

Alex Schladebeck - Alemania

Cuando comencé, el rol del evaluador en un equipo (si el evaluador era parte del equipo) a menudo era ser el único responsable de la automatización de la interfaz de usuario y las pruebas manuales. En los últimos 12 años, he visto una enorme diversificación del rol, y siempre de una manera dependiente del contexto según cómo operaba el equipo. Veo evaluadores trabajando en más tareas de automatización, incluso emparejándose a nivel de unidad con desarrolladores. Los veo involucrados en todos los puntos del proceso. Los veo organizando sesiones de prueba con el equipo para realizar pruebas exploratorias.

Capítulo 11: El nuevo rol de un probador

Los veo haciendo campaña para mejorar los circuitos de retroalimentación. Incluso he visto a evaluadores comenzar a corregir errores o implementar funciones.



Haz conexiones

Para mí, esta diversificación y confusión de roles es una enorme libertad y una gran responsabilidad. Significa que tenemos que preguntarnos: "¿Cómo encajan mejor yo, mis habilidades y mi potencial para aprender en el contexto de este equipo?" Nos convertimos en un "pegamento de calidad y comunicación", identificando y llenando los vacíos en cualquier equipo.

Empecé a utilizar el término "ingeniero de calidad integrado" o "consultor de calidad integrado" para este puesto. El problema con el título "probador" es que contiene el nombre de una de las muchas actividades que realizamos, por lo que escuchas preguntas como: "Si todos participan en las pruebas, ¿por qué necesitamos un probador?" o declaraciones como: "Los desarrolladores están automatizando pruebas, por lo que no necesitamos una función de tester".

Las pruebas son solo una de las muchas cosas que hace un evaluador. En mi opinión, debemos luchar contra la idea de que un equipo ágil debería estar formado por "quimeras": una combinación de diferentes roles, o un miembro del equipo con una navaja suiza que puede hacer todo: requisitos, UX, pruebas, seguridad, frontal y posterior. Los equipos ágiles deben ser diversos y multifuncionales, y eso significa que necesitamos personas con diferentes orígenes, intereses y especializaciones, sin que ninguna persona sea un silo o un cuello de botella. Creo que es un equilibrio que se puede lograr.

Considero que el rol del evaluador consta de múltiples actividades. Hay cosas que siempre han sido parte del rol, como trabajar con las partes interesadas, aportar experiencia a las actividades de prueba, colaborar con desarrolladores y otros evaluadores, apoyar al propietario del producto,

organizar y adaptar la estrategia general de calidad, obtener buenos datos de prueba e identificar riesgos. Creo que habrá aún más tareas que los evaluadores podrán asumir o respaldar en el futuro. Algunos de estos podrían ser: trabajar con el equipo para garantizar la capacidad de prueba y la observabilidad de las pruebas y el monitoreo en producción, hacer preguntas sobre el sistema de producción para explorar cómo se está utilizando, perfeccionar nuestro desempeño y enseñar habilidades para pruebas exploratorias, ayudar al equipo a centrarse en el valor (y a veces en el minimalismo, es decir, el valor de algo no hecho). También veo que los evaluadores comienzan a agregar "salud del equipo" a sus atributos de calidad, prestando atención a los niveles de comunicación y estrés del equipo en su conjunto. Después de todo, estas son cosas que pueden afectar en gran medida la calidad.

En resumen, creo que el papel del tester sigue siendo importante. Es la persona del equipo cuya principal prioridad es la calidad. También son personas apasionadas por la calidad y las pruebas, y las defienden y defienden.

El recorrido profesional de un tester ágil

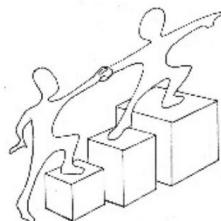
Pablo Carvalho – Canadá

Cuando entreno equipos ágiles, ayudo a todo el equipo a aprender a trabajar juntos y aprovechar las fortalezas de los demás. El propietario de un producto aporta conocimientos empresariales y de la industria, un programador aporta sólidas habilidades de codificación y desarrollo, un diseñador aporta información sobre la perspectiva y la experiencia del usuario y un evaluador también aporta algo de valor.

El recorrido profesional de un evaluador comienza al alejarse de las pruebas accidentales y aleatorias y pasar a un diseño cuidadoso de las pruebas a través de modelos, técnicas y otras habilidades y conocimientos especializados.

Agile pone un fuerte énfasis en trabajar en estrecha colaboración con otros, por lo que los evaluadores deben salir de sus cabezas cuando piensan en realizar pruebas y encontrar

su voz para ayudar a otros miembros del equipo a comprender las muchas formas de generar información de calidad sobre los sistemas en desarrollo. Un gran evaluador ágil pasa más tiempo con un marcador de pizarra en la mano y colaborando con otros miembros del equipo que cualquier otra cosa. Se trata de ayudar al resto del equipo a ver y comprender más sobre el sistema, antes de construirlo. "Build Quality In" es más que un eslogan; es una realidad que los grandes evaluadores pueden ayudar a habilitar en equipos colaborativos de alto rendimiento.



Mentor

El recorrido profesional de un evaluador ágil avanza desde pruebas aleatorias y desordenadas hasta la comprensión de las técnicas y el diseño de buenas pruebas, hasta encontrar una voz y expresar estas ideas para que puedan ayudar a elevar el desempeño del resto de su equipo.

Como evaluador, intente mejorar y aumentar la comprensión de todos sobre los sistemas y soluciones a través de una exploración cuidadosa. Si deja de lado la noción de que debe ser usted quien escriba casos de prueba o automatización de programas, piense en dónde sus habilidades y conocimientos únicos pueden ayudar a impulsar a su equipo.

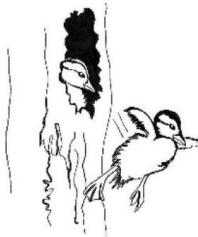
El fascinante camino de evolucionar como probadores

Claudia Badell – Uruguay

Como testers, podemos contribuir y agregar valor desde diferentes perspectivas: como facilitadores y evangelizadores hacia las pruebas y la calidad en un equipo de producto, como coaches, como consultores de pruebas, como expertos en ciertos tipos de pruebas (pruebas de usabilidad, pruebas de accesibilidad, pruebas de seguridad, testing, pruebas de rendimiento, entre otros), y más. Los evaluadores ya no son vistos como guardianes de la calidad, por lo que podemos ser vistos y valorados como defensores de la calidad.

Hoy en día, es cada vez más frecuente que los testers formen parte del equipo de desarrollo y realicen aportaciones desde el inicio del proceso de desarrollo. En mi experiencia, el papel del tester en este contexto está evolucionando. Además de realizar actividades de prueba para respaldar las pruebas manuales y las comprobaciones automatizadas, los evaluadores colaboran para construir puentes dentro del equipo con el fin de alcanzar un entendimiento y un compromiso común sobre las pruebas. También definen, dan seguimiento y ajustan las estrategias de testing a aplicar por todo el equipo. Además, comparten y evangelizan sus conocimientos sobre pruebas dentro del equipo.

A medida que la tecnología, las metodologías y los procesos evolucionan y los equipos y las comunidades maduran, creo que es importante que tengamos una actitud proactiva para adaptarnos a dichos cambios. El futuro traerá nuevos desafíos y oportunidades. Dependiendo del contexto, es posible que se necesiten diferentes habilidades y diferentes actividades de prueba, pero en mi experiencia, existen habilidades básicas que son necesarias para mantener el ritmo del desarrollo de software.



Experimenta y encuentra nuevas oportunidades.

Estas habilidades son:

- estar ansioso por aprender y probar nuevos experimentos para mejorar las pruebas estratégicas en el equipo.
- ser un excelente formulador de preguntas durante todo el ciclo de vida del producto. Dependiendo del tipo de información que recopilemos, las preguntas se pueden formular de diferentes maneras. Se pueden hacer verbalmente o por escrito, por lo que es importante tener habilidades de comunicación claras y bien estructuradas. También se pueden realizar mediante programación; Por ejemplo, si quisieramos verificar ciertas respuestas entre dos servicios, las habilidades técnicas son importantes.
- tant.
- tener habilidades de modelado como una manera de entender qué probar y definir estrategias de prueba que cubran los diferentes aspectos que se necesitan.
- tener cierto grado de conocimiento técnico para colaborar en la definición de los aspectos de capacidad de prueba de la solución mientras se desarrolla el software, por ejemplo, para soportar pruebas unitarias y pruebas de integración automatizadas.
- tener una actitud de compartir y colaborar.

Estamos en un momento apasionante en el que podemos dar forma a parte de nuestro futuro.
¿Cómo te estás preparando para ello?

Se todo lo que puedas ser

Mike habla - Nueva Zelanda

Durante los últimos seis años, mi equipo de prueba pasó de ser un solo grupo que trabajaba en un proyecto en cascada a la vez a personas que trabajan en varios equipos.

Se habla mucho sobre un enfoque de equipo completo para la calidad y las pruebas, pero se busca que los evaluadores, como especialistas, lideren este espacio.

Eso significa crear un enfoque de primer paso en una nueva historia o característica, pero también es importante facilitar una discusión con el equipo más grande sobre estos enfoques para obtener comentarios y explorar el enfoque.

También significa que si una tarea de prueba es demasiado onerosa para que los evaluadores la completen solos, pueden ayudar a organizar una división del trabajo entre los miembros dispuestos del equipo.

Considero que un candidato frecuente para esto son las pruebas entre navegadores y dispositivos. A menudo probamos historias en la iteración utilizando nuestros dispositivos principales, pero ocasionalmente visitamos nuestro producto en una gama mucho más amplia de artículos. Aquí es donde el equipo y sus nuevos ojos pueden ayudar, y un poco de organización por parte del evaluador puede marcar una gran diferencia.

Aunque la mayoría de las conversaciones sobre probar un producto ocurren dentro de su equipo, también es útil "ponerse al día" con otras personas en las mismas disciplinas para compartir ideas y lo que ha estado funcionando en otros equipos.

Esto también puede convertirse en tutoría para ayudar a las personas a lidiar con problemas específicos y a volverse más atrevidas para probar nuevas ideas.



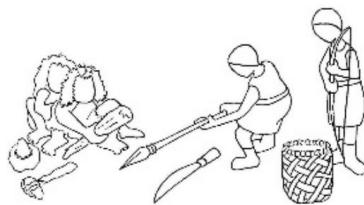
Se todo lo que puedas ser

Comience con una conversación

Kathleen Naughton - Estados Unidos

Las pruebas de software han evolucionado y se han estancado simultáneamente. Ha evolucionado hasta el punto de que algunas organizaciones han reducido el número o incluso eliminado los evaluadores de sus equipos. Se ha estancado porque estas mismas organizaciones han luchado por valorar las habilidades especializadas que un evaluador aporta a los equipos.

En mi experiencia, cuando se redujeron o eliminaron los evaluadores, los equipos intentaron completarlo acercándose a las prácticas de TDD para tener una gran cantidad de pruebas unitarias automatizadas. Intentan realizar algunas pruebas dentro del equipo (probar el código de cada uno) y dependen en gran medida de su canal de entrega continua para ejecutar sus pruebas automatizadas. Lo que a menudo se acaba pasando por alto son las pruebas de integración y experiencia de usuario que son esenciales para productos de alta calidad. Si hay evaluadores en estas organizaciones, están presentes para realizar pruebas manuales una vez finalizado el código. Cualquier idea o sugerencia hecha por estos evaluadores tiende a perder prioridad en la cartera de pedidos del producto, aplazando el desarrollo de funciones.



Sea relevante e influya en quienes le rodean

Creo que una habilidad esencial que los evaluadores necesitan para ser relevantes es poder leer y comprender el código. Esto les permite comprender qué pruebas unitarias u otras pruebas automatizadas están verificando y permite identificar los vacíos de prueba que el evaluador puede llenar. También permite reducir la superposición de pruebas. Si ya hay unidad o integración

pruebas presentes, el enfoque de prueba puede estar más dirigido a las actividades del usuario final que no están cubiertas. Otra habilidad esencial creo se necesita podría considerarse una habilidad blanda. La habilidad de tener conversaciones con programadores sobre sus pruebas unitarias y de integración es poderosa. Estas conversaciones pueden influir en las decisiones de diseño que, a su vez, permiten obtener resultados de mayor calidad. Aportar conocimientos sobre cómo tener conversaciones cruciales permite a todo el equipo producir un mejor software.

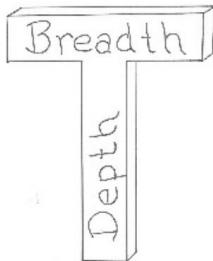
El mundo no necesita más damas

[Aldo Rall](#) - Nueva Zelanda

Las pruebas han evolucionado a lo largo de los años y la industria ha desarrollado habilidades y prácticas de ingeniería de pruebas, aunque no de manera fluida. Creo que hemos entrado en una era de iluminación sobre las pruebas y hay un gran cambio en la forma en que las organizaciones y los evaluadores piensan ahora sobre el papel de las pruebas.

Observo un movimiento creciente hacia la importancia de las habilidades. Cuantas más habilidades tenga un individuo, más valioso se volverá para un equipo u organización. Aquellas personas con habilidades que van más allá de las habilidades de ingeniería de pruebas son las que pueden contribuir más que alguien que tiene un conjunto básico de habilidades de diseño y ejecución de pruebas. Esta idea se enfatiza en el pensamiento sobre la generalización de especialistas, tal como lo analiza Scott Ambler (<http://www.agilemodeling.com/essays/generalizingSpecialists.htm>). o Habilidades en forma de T como lo comentan Lisa y Janet en sus libros. Si desea preparar su carrera para el futuro, desarrolle sus habilidades de ingeniería de pruebas, así como habilidades fuera del mundo tradicional de las pruebas. Olvídate de los títulos; dentro de diez años no significará mucho para nadie que le llamen "ingeniero de pruebas", "especialista en pruebas", "probador", "analista de pruebas" o "ingeniero de verificación". En última instancia, las habilidades son más

valiosos que los títulos de trabajo recopilados; Ciertamente he descubierto que eso es cierto en mi propia carrera.



habilidades en forma de T

Yo consideraría un enfoque holístico caracterizado por conversaciones inclusivas sobre el "y". El mejor ejemplo que se me ocurre es combinar tu colección de habilidades de maneras únicas que se adapten a tu contexto específico, sabiendo que incluso eso cambiará. ¿Cómo se pueden combinar un conjunto de habilidades de análisis e ingeniería de pruebas en una situación determinada? ¿Cómo se pueden combinar un conjunto de habilidades de negociación con habilidades de enseñanza? ¿Cómo se pueden combinar diferentes habilidades de ingeniería de pruebas para obtener una mejor cobertura de las pruebas? Piense en "y".

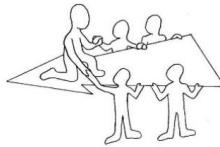
Creo que una de las habilidades clave que debe tener un profesional moderno es la capacidad de comprender un contexto, comprender su naturaleza cambiante y adaptarse en consecuencia. Los verdaderos maestros del futuro lugar de trabajo serán aquellos que sean capaces de observar un contexto, aplicar la combinación de habilidades adecuada al propósito y luego ajustar continuamente la combinación de habilidades a medida que el contexto evoluciona. Se necesita tiempo para desarrollar esa intuición y es un ámbito en constante cambio. Aprender nuevas habilidades enriquecerá la capacidad y el valor que dicha persona aporta a las organizaciones y equipos.

Aportamos muchas habilidades diferentes además de simplemente evaluar las habilidades de ingeniería. Me gustaría sugerir que consideremos (tomando prestado descaradamente de otros) un enfoque multifacético. Llámelo "Habilidades holísticas de pruebas ágiles" o "Los diez sombreros para pensar de las pruebas ágiles", o cualquier otra cosa que tenga sentido para usted. Algunas de estas facetas podrían ser:

- Consultor: Sí, a veces tendremos que “consultar” dentro de nuestro equipo o con otro equipo para ayudar a resolver problemas y cuestiones.
- Especialista en ingeniería de pruebas: debemos diferenciar nuestras pruebas de cebollas de chalotes. Requerimos buenas y sólidas habilidades de ingeniería de pruebas. • Agile Scholar: sigue estudiando y aprendiendo sobre Agile, trae ideas al equipo y experimentar.
- Entrenador: Tenemos grandes oportunidades para entrenar al equipo o incluso a compañeros de trabajo (dentro y fuera del equipo). Esta es una habilidad para la vida, en mi opinión.
- Mentor: A veces tenemos que asesorar a alguien en las pruebas. • Facilitador: A veces sólo necesitamos asumir el papel de facilitador para una decisión, discusión, explicación, etc. • Agente de cambio: A veces podemos provocar cambios y agitación, abogar por una nueva práctica/técnica/método para experimentar con y aprender de. • Líder: Sí, es posible que a veces se nos solicite que demos un paso al frente y desempeñemos un liderazgo en nombre del equipo. • Profesor: Eso es evidente, especialmente si hay escasez de habilidades para realizar pruebas en el equipo/organización. • Académico en el ámbito empresarial/defensor del sentido común/pensador general: a veces es bueno alejarse y ver el bosque desde los árboles.

Los pensamientos de Lisa y Janet.

Esperamos que haya disfrutado leyendo los pensamientos de otras personas sobre lo que consideran el papel de un evaluador. Ahora compartiremos el nuestro. Hemos alentado a los evaluadores a ayudar a sus compañeros de equipo que no son evaluadores a aprender habilidades de prueba. Cuando todo el equipo asume la responsabilidad de la calidad y las pruebas, cada miembro del equipo necesita cierta base en las habilidades de prueba. Para lograr esto, hemos identificado habilidades que recomendamos que los especialistas en pruebas aprendan.



Usa tus habilidades de pensamiento

- habilidades de colaboración para participar activamente en prácticas como mapas mentales o mapas de ejemplo
- Habilidades de facilitación para ayudar a los miembros del equipo a comunicarse mejor, facilitar reuniones como retrospectivas, facilitar talleres para ayudar a los no evaluadores a aprender habilidades de prueba. •
- Habilidades de enseñanza para compartir sus conocimientos con otros equipos. miembros
- Habilidades de entrenamiento para ayudar al equipo a identificar problemas y diseñar experimentos para mejorar.
- habilidades de comunicación para dar y recibir retroalimentación de manera efectiva (consulte el Capítulo 4, "Habilidades de pensamiento para las pruebas", en Más Pruebas ágiles para obtener información adicional)

Vemos una necesidad creciente de que los evaluadores actúen como consultores de pruebas para sus equipos. Muchos equipos tienen una proporción baja de evaluadores dedicados a desarrolladores y otras funciones que no son de prueba. Los evaluadores podemos agregar más valor ayudando a todos a tener las competencias necesarias para realizar actividades de prueba esenciales. Todos los miembros del equipo pensarán más en las pruebas y serán más conscientes de la necesidad de generar calidad desde el principio.

Capítulo 12: Ingredientes para el éxito

Cada equipo de entrega de software ágil recorre su propio viaje de aprendizaje. Nuestro objetivo es mejorar continuamente nuestra capacidad de ofrecer valor a nuestros clientes con frecuencia, manteniendo al mismo tiempo el estándar de calidad deseado de nuestro negocio. Cada equipo hace esto con una combinación única de dominio empresarial, producto de software, pila de tecnología, marcos y prácticas.

A lo largo de los años, hemos descubierto que entre todas estas diferencias, ciertos ingredientes para el éxito benefician a todos los equipos.

Factores de éxito

En nuestro primer libro, Agile Testing, nuestro capítulo de resumen comprendía siete factores de éxito que pensamos que eran necesarios (aunque no suficientes) para tener éxito en la entrega de un producto de calidad. Es fácil sentirse abrumado al planificar y ejecutar actividades de prueba durante ciclos de entrega cortos. A continuación se muestra una breve lista de factores clave de éxito y prácticas de pruebas ágiles básicas para guiar a sus equipos.

“Enfoque de equipo completo”

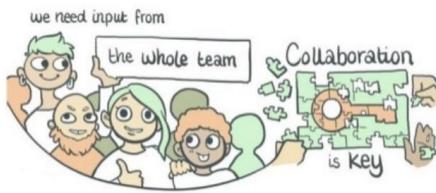
Elisabeth Hendrickson nos enseñó que "las pruebas son una actividad, no una fase". Las pruebas son una parte integral del desarrollo de software, junto con la codificación y muchas otras actividades. Con esta perspectiva, es fácil para todos ayudar con las tareas de prueba según sea necesario.

Los evaluadores pueden enseñar habilidades a otros miembros del equipo, como obtener ejemplos concretos de comportamiento deseado y no deseado de expertos en negocios, evaluar diferentes atributos de calidad o realizar pruebas exploratorias.

Los programadores pueden ayudar a los evaluadores a comprender la arquitectura del sistema para realizar mejores pruebas o incluso enseñarles construcciones básicas de codificación.

Cada miembro del equipo puede transferir algunas de sus habilidades profundas a otros miembros del equipo, independientemente de su función.

Cuando los equipos se dan cuenta de que las pruebas y la calidad son un problema de equipo, pueden incorporar sus diversas habilidades y desarrollar una atmósfera de confianza y seguridad, así como crear un entorno de aprendizaje donde puedan experimentar y mejorar continuamente.



Dibujo de Constanza Ermitaño

Mentalidad de prueba ágil

Los evaluadores ya no son la “policía de la calidad” que determina las decisiones de “aceptar o no”. Los evaluadores o miembros del equipo que realizan actividades de prueba pueden explicar los riesgos e impactos de los resultados de las pruebas para que la empresa pueda tomar una decisión informada sobre el lanzamiento a producción.

Como miembro del equipo con una mentalidad de prueba ágil, significa que eres curioso y quieres aprender más sobre todo para ayudarte a hacer tu trabajo. Significa que aplicas principios y valores ágiles.

Significa colaborar con los miembros del equipo técnico y comercial, teniendo en cuenta el panorama general al combinar pequeños incrementos de funciones. Usted se centra en la prevención de errores, por lo que no tiene que dedicar tanto tiempo a buscar errores más adelante.



Dibujo de Constanza Ermitaño

Automatiza tus pruebas de regresión

Hay algunas cosas que debe recordar cuando su equipo comience a automatizar. Es un problema de equipo, así que piense en “todo el equipo” y colabore para automatizar en todos los niveles. Los programadores son buenos escribiendo código, los evaluadores son buenos especificando pruebas y las personas con otras habilidades especializadas en el equipo pueden ayudar con los datos de prueba, la infraestructura y más. La pirámide de automatización de pruebas es un buen modelo visual para formar y hacer evolucionar la estrategia de automatización del equipo. Al mantener las pruebas simples y fáciles de mantener, un equipo puede trabajar para tener suficientes pruebas de regresión que les den confianza a la hora de realizar el lanzamiento.

La automatización de pruebas es una verificación para garantizar que no se haya olvidado de cambiar algo, es decir, es un detector de cambios. Una buena estrategia de automatización le brinda tiempo para realizar pruebas exploratorias para encontrar problemas antes de que lo haga su cliente.

Proporcionar y obtener comentarios

El desarrollo exitoso de software depende de una retroalimentación rápida. Los equipos necesitan saber de inmediato si un cambio ha provocado una falla no deseada. Quieren saber cómo reaccionan los clientes ante una nueva función. Los probadores son fundamentales para crear y continuar acortando los distintos

bucles de retroalimentación, incluida la creación de pruebas automatizadas, la participación en pruebas exploratorias y la observación del uso de producción para aprender cómo los clientes usan el producto.



Las personas también necesitan retroalimentación para sí mismas para poder encontrar más formas de agregar valor. Las habilidades de escucha y observación son clave.

Sugerencia: mientras colabora con otros miembros del equipo, pregúntale qué vacíos puede llenar y cómo puede contribuir de manera más efectiva.

Construir una base de prácticas básicas

Existen prácticas básicas que han demostrado ser eficaces para ayudar a los equipos a incorporar calidad en su producto.

- Cada equipo necesita una integración continua para entregar software exitosamente con una cadencia frecuente a lo largo del tiempo. Cada vez que un miembro del equipo realiza un cambio en el repositorio de control del código fuente, debe iniciar un proceso de compilación que integre todos los cambios de código y los verifique con pruebas automatizadas. Cada equipo tiene un proceso de implementación para crear una versión candidata e implementarla en un entorno de prueba o producción, incluso si incluye etapas manuales.
- Muchos equipos todavía luchan por tener entornos de prueba confiables que se parezcan lo más posible a la producción y les permitan controlar fácilmente qué versión de compilación se implementa. de hoy

La infraestructura en la nube proporciona aún más opciones para crear entornos de prueba temporales para probar una versión de compilación específica e implementar automáticamente nuevas versiones en entornos de prueba permanentes.

- La deuda técnica es como la deuda de tarjetas de crédito; continúa creciendo si sólo se paga la cantidad mínima o parte de los intereses.

La deuda técnica continúa creciendo si el equipo no se toma el tiempo para refactorizar el código, crear pruebas de regresión automatizadas adecuadas, actualizar los marcos y administrar otra infraestructura necesaria. Con el tiempo, incluso el cambio de código más pequeño plantea grandes riesgos y requiere mucho tiempo dedicado a tareas de prueba manuales. Invierte tiempo en gestionar la deuda técnica en el código y en tus pruebas automatizadas.



- Los cambios pequeños y frecuentes generalmente son menos riesgosos que los grandes e infrecuentes. Hay menos cosas que pueden salir mal y las fallas se pueden diagnosticar rápidamente. Los equipos que dividen las grandes ideas de funciones en pequeños "lanzamientos de aprendizaje" e incluso en historias más pequeñas tienen más probabilidades de ofrecer lo que sus clientes desean en el momento oportuno. manera.
- La codificación y las pruebas son parte de un mismo proceso. Este es el núcleo del enfoque de todo el equipo en materia de pruebas y calidad. Las pruebas y la codificación ocurren juntas, de la mano, desde la idea de la función hasta la evaluación de la función en producción. • La sinergia entre prácticas surge de realizar todas estas prácticas centrales juntas. El desarrollo basado en pruebas, la propiedad colectiva del código y la integración continua garantizan coherencia y una retroalimentación rápida. La refactorización depende de tener automatizada

pruebas de regresión. Las prácticas ágiles están probadas y son verdaderas y están diseñadas para realizarse en conjunto.

Colaborar con los clientes

Los evaluadores hablan el idioma de dominio de las partes interesadas del negocio y el lenguaje técnico de los miembros del equipo de entrega. También es importante reunir a las personas adecuadas cuando es necesario conversar sobre cómo debe comportarse una característica o cómo debe verse un diseño.

Los evaluadores pueden ayudar a los propietarios de productos a articular reglas comerciales para cada historia e ilustrarlas con ejemplos concretos. Esta es una de las formas más valiosas en que los evaluadores contribuyen en los equipos.

Mira la imagen completa

Si bien los equipos ágiles se centran en pequeños cambios y pequeñas porciones de funciones en un momento dado, también deben tener en cuenta el panorama general. Los evaluadores tienen talento para identificar qué partes del sistema podrían verse afectadas por un pequeño cambio en particular y tienen la perspectiva del cliente.

El modelo ágil de cuadrantes de prueba contribuye en gran medida a ayudar al equipo a tener en cuenta el panorama general al planificar las actividades de prueba.

Las pruebas exploratorias son un ejemplo de una actividad de prueba que puede descubrir consecuencias inesperadas de una nueva capacidad de aplicación.

Tenemos buenas formas de analizar cómo los clientes utilizan nuestro producto. Todo esto nos ayuda a centrarnos en ofrecer el valor adecuado.

Prácticas de fomento de la confianza

En nuestro segundo libro, *More Agile Testing*, identificamos algunas prácticas de prueba básicas que ayudan a los equipos a generar la confianza necesaria para lanzar cambios en producción con frecuencia. Estas prácticas son especialmente importantes a medida que más equipos avanzan hacia la entrega o implementación continua.

Ejemplos de uso

Los ejemplos concretos de cómo debe comportarse la capacidad de una aplicación ayudan a todos los miembros del equipo a comprender las reglas comerciales. Estos ejemplos se pueden convertir en pruebas que guíen el desarrollo. Se pueden automatizar para que el equipo sepa cuándo ha terminado con una historia o característica. Las pruebas automatizadas pasan a formar parte de conjuntos de pruebas de regresión que proporcionan información rápida sobre si un nuevo cambio ha afectado el comportamiento de producción existente. Los ejemplos ayudan a los equipos a mantener el rumbo.



Prueba exploratoria

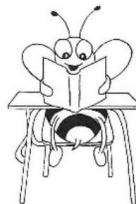
La automatización de las pruebas de regresión deja más tiempo para las pruebas exploratorias, una de las mejores formas de encontrar las "incógnitas desconocidas" que podrían causar graves fallos de producción. Los programadores pueden aprender a realizar pruebas exploratorias en cada historia antes de considerar que su trabajo está "terminado". Este es otro ciclo de retroalimentación rápida. Todos los miembros del equipo pueden aprender y utilizar habilidades de pruebas exploratorias. No sólo identificarán problemas inesperados, sino que también encontrarán capacidades faltantes que alimentarán nuevas ideas de funciones.

Prueba de funciones

Es esencial realizar pruebas en todos los niveles de detalle. Debido a que los equipos ágiles se centran en la historia, deben recordar realizar pruebas también a nivel de funciones. Una parte importante de esto es identificar qué característica realmente debe incluir. Los evaluadores pueden ayudar a descubrir qué es valioso para los clientes haciendo preguntas sobre qué debería dejar de lado la empresa para ofrecer otras funciones de gran valor.

Aprendizaje continuo

El éxito del equipo depende de la seguridad psicológica, la confianza y el tiempo para aprender. El equipo debe trabajar en conjunto para identificar el mayor obstáculo para brindar el nivel de calidad deseado, ya sea una automatización de pruebas inadecuada, comentarios que llevan demasiado tiempo o la creación de funciones que nadie quería. Luego podrán diseñar pequeños experimentos para empezar a superar ese obstáculo. Los evaluadores pueden ayudar al equipo a aprender a generar calidad transfiriendo habilidades de prueba. Otros miembros del equipo pueden ayudar a los evaluadores a mejorar sus habilidades en forma de T para que puedan contribuir de más maneras.



Sensibilidad al contexto

Cada equipo trabaja dentro de su contexto único. El tamaño de la empresa, el ámbito empresarial y su entorno regulatorio, la tecnología involucrada, las necesidades de infraestructura: estas son sólo algunas consideraciones para un equipo que considera cómo mejorar su capacidad de ofrecer valor a los clientes con frecuencia. No adopte una herramienta o práctica porque es lo que hace Google o Facebook; utilice lo que sea apropiado para su contexto.



Se realista

Los evaluadores se destacan por brindar comentarios. Puede resultar difícil dar malas noticias. Pero es importante permanecer anclado en la realidad. Si un cambio es riesgoso y el equipo no ha mitigado adecuadamente ese riesgo con pruebas y otras actividades, las partes interesadas del negocio deben saberlo.

Los evaluadores pueden actuar como consultores para ayudar a todos los miembros de su equipo a mejorar sus habilidades de prueba y poder hacer visibles las inquietudes sobre la calidad para la empresa. Cuando el equipo experimente cuellos de botella con las pruebas, hágalo visible, conviértalo en un problema de equipo a resolver. Puede resultar tentador pasar por alto los problemas para mantener contentos a los ejecutivos de la empresa, pero no estarán contentos si los clientes experimentan problemas.

También es importante para la moral del equipo saber que pueden decir que no. Por ejemplo, "No, no podemos aceptar más historias" o "No, no podemos agregar una nueva historia a menos que elimines una de las otras". ¡Se realista!

Caminos hacia el éxito

Sabemos que hay muchos evaluadores y equipos que se sienten estresados, especialmente en equipos que aún están en transición hacia el uso de valores, principios y prácticas de desarrollo ágil. Por ejemplo, la dirección aún no ha descubierto cómo debe cambiar su función y puede exigir entregas más frecuentes e imponer plazos poco realistas. Las pruebas aún deben realizarse y, en demasiados casos, los evaluadores aún tienen la responsabilidad total de todas las actividades de prueba. Nos gustaría pensar que existe alguna herramienta mágica que resolverá todos nuestros problemas, ¡pero sabemos que eso no es la realidad!

Convertir los problemas de prueba en problemas que todo el equipo de entrega debe abordar es vital para aprender cómo incorporar calidad en sus productos y lograr un éxito sostenible. Estos factores clave de éxito y prácticas de fomento de la confianza proporcionan un marco para ayudar al equipo a decidir sus próximos pasos en el camino hacia la mejora.

Según nuestra experiencia, un equipo de entrega tarda años en alcanzar el nivel deseado de rendimiento y calidad. Podríamos agregar un octavo factor clave de éxito: ¡la paciencia! Las retrospectivas frecuentes del equipo (recomendamos al menos una por semana para los equipos nuevos) también son clave para identificar el mayor problema relacionado con la calidad y diseñar un pequeño experimento para comenzar a solucionarlo. Las diversas habilidades y experiencia en un equipo multifuncional hacen que resolver esos problemas sea mucho más fácil.

Un ejemplo

El equipo está frustrado porque el propietario del producto rechaza un alto porcentaje de las historias que ofrece. El constante retrabajo, a veces días después de que el equipo pensaba que la historia estaba “terminada”, los está frenando. El tiempo del ciclo (el tiempo desde que empiezan a trabajar en una historia hasta que se implementa en producción) es mucho más largo de lo que les gustaría. ¿Qué factor clave de éxito puede ayudar?

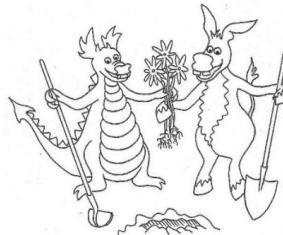
El enfoque de todo el equipo es obvio. Reunamos a todo el equipo, o a un grupo representativo que incluya todos los roles, para discutirlo. ¿Qué práctica de fomento de la confianza ayudaría? Cuando el propietario del producto rechaza una historia, normalmente es porque el comportamiento de esa parte de la aplicación no es el que ella quería. El equipo entendió mal los requisitos. El equipo está aprendiendo continuamente (una de las prácticas que fomentan la confianza) y uno de los evaluadores acaba de aprender sobre el mapeo de ejemplos. Deciden experimentar con mapas de ejemplo para ver si generarán una mejor comprensión compartida de cada historia. Su hipótesis es que el mapeo de ejemplo reducirá la tasa de rechazo de historias en un 20 % durante las próximas dos semanas, lo que resultará en un ahorro del 10 % en el tiempo promedio del ciclo.

Miden y experimentan para ver si su hipótesis es cierta.

En este ejemplo real, el experimento fue un éxito. Se superaron los objetivos de reducción de la tasa de rechazo y del tiempo de ciclo. En dos meses, la tasa de rechazo y el tiempo del ciclo se redujeron en un 50 %. El equipo encontró más beneficios del mapeo de ejemplo, ya que

ayudó a especificar escenarios para guiar el desarrollo en forma de pruebas de desarrollo basadas en el comportamiento. Pero si el experimento hubiera fracasado, el equipo habría diseñado otro experimento, guiado por los factores de éxito y las prácticas de fomento de la confianza.

Cuando nuestros propios equipos se sienten estancados en un problema, recurrimos a los factores clave de éxito y a las prácticas de fomento de la confianza, junto con los diez principios para las pruebas ágiles del Capítulo 1, para ayudarnos a planificar nuestros próximos pasos. Nos guiarán a lo largo de nuestro viaje de aprendizaje a medida que mejoramos nuestra capacidad de lograr cambios pequeños y valiosos para nuestros clientes de manera frecuente y sostenible.



Gracias por leer y esperamos que tenga éxito en su propio viaje.

Glosario

Pruebas ad hoc: una actividad de prueba informal en la que se buscan errores de forma no estructurada sin ningún plan previo.

Diagrama de contexto: diagrama de alto nivel que representa todas las entidades externas que pueden interactuar con un sistema, incluidos otros sistemas, entornos y actividades.

Cliente: Extreme Programming (XP) utiliza el término "cliente" para referirse a una parte interesada del negocio, una persona del producto o un usuario final que se reúne con el equipo de programación para establecer prioridades, responder preguntas y tomar decisiones sobre el comportamiento de las funciones. En los equipos ágiles contemporáneos, el término puede representar a todas y cada una de las partes interesadas del negocio, miembros del equipo de producto, usuarios finales y cualquiera que ayude a guiar el desarrollo y acepte las historias entregadas.

Final del juego: El final del juego es el momento antes del lanzamiento cuando el equipo de entrega aplica los toques finales al producto. No es un período de corrección de errores o de "endurecimiento", sino una oportunidad de trabajar con grupos ajenos al desarrollo para ayudar a llevar el software a producción. Ejemplos de actividades finales incluyen pruebas adicionales de migraciones e instalación de bases de datos.

Iteración: cuadro de tiempo utilizado para la planificación, con la intención de que al final haya un "producto potencialmente entregable". El término Scrum para esto es "sprint". La planificación en plazos de dos semanas es una práctica común hoy en día, incluso en equipos que realizan entregas continuas e implementan en producción con más frecuencia.

Kanban: Un enfoque de planificación derivado de la fabricación Lean en el que los equipos trabajan de forma basada en flujos. Utilizan límites de trabajo en progreso (WIP), incorporando nuevas historias que están "listas" para llenar un espacio WIP recién vacío. El equipo planifica, según sea necesario, algunas historias nuevas a la vez.

Versión de aprendizaje: las primeras versiones entregadas a un cliente con el fin de obtener comentarios para aprender y realizar ajustes (https://medium.com/@Ardita_K/the-learning-release-70374d2450b3).

Mapa mental: diagrama visual que se utiliza como herramienta de lluvia de ideas, especialmente cuando varias personas colaboran a la vez. Comienza con un concepto, idea o tema en el nodo raíz, con ideas conectadas al nodo raíz y entre sí a medida que se generan. Los mapas mentales pueden funcionar bien para la planificación de pruebas y otras actividades.

Emparejamiento (programación en pares, pruebas en pares): dos personas trabajando lado a lado en la misma estación de trabajo, preferiblemente con dos monitores reflejados, dos teclados y dos ratones, para escribir código de producción o prueba o realizar otras actividades de prueba. Tener dos personas, cada una con una perspectiva y un conjunto de habilidades diferentes, ayuda a detectar los problemas de inmediato y alcanzar mejores soluciones. En el emparejamiento de estilo fuerte, una persona, el navegante, es libre de observar y sugerir ideas, mientras que la otra persona actúa como "conductor"; los roles de conductor/navegador cambian con frecuencia.

Diagrama de estado: una técnica visual utilizada para dar una descripción abstracta del comportamiento²² de un sistema²³ en respuesta a diversos acontecimientos. Este comportamiento se analiza y representa como una serie de eventos que pueden ocurrir en uno o más estados posibles.

Desarrollo basado en pruebas (TDD): en el desarrollo basado en pruebas, el programador escribe y automatiza una pequeña prueba unitaria, que inicialmente falla, antes de escribir la cantidad mínima de código que hará que la prueba pase. El código se refactoriza según sea necesario para cumplir con estándares aceptables. El código de producción está diseñado para funcionar una prueba a la vez. TDD, también conocido como diseño basado en pruebas, es más una práctica de diseño de código que una actividad de prueba y ayuda a crear código sólido y fácil de mantener.

22<https://en.wikipedia.org/wiki/Behavior>

23<https://en.wikipedia.org/wiki/System>

Recursos para más

Aprendiendo

General

Pruebas ágiles: una guía práctica para evaluadores y equipos ágiles, y pruebas más ágiles: viajes de aprendizaje para todo el equipo, Lisa Crispin y Janet Gregory, <https://agiletester.ca>²

Bibliografía de More Agile Testing, digitalizada por Kristine Corbus

- <https://testretreat.com/2018/01/28/more-agile-testing-introduction/> • <https://testretreat.com/2018/01/29/more-agile-testing-learning-better-testing/> • <https://testretreat.com/2018/01/29/more-agile-testing-planning/> • <https://testretreat.com/2018/01/30/more-agile-testing-business-valor>
- <https://testretreat.com/2018/02/15/more-agile-testing-test-automation/>

Obtener una comprensión compartida - Colaboración

Descubrimiento: explore el comportamiento utilizando ejemplos, Seb Rose y Gáspár Nagy, 2017, <http://bddbooks.com/>

Mapeo de historias de usuario: creación de mejores productos utilizando el diseño de software ágil, Jeff Patton, O'Reilly Media, 2014.

² <https://agiletester.ca/>

Mapeo de impacto, Gojko Adzic, <http://impactmapping.org>

“Experimento con mapeo de ejemplo”, <https://lisacrispin.com/2016/06/02/experiment-example-mapping/>

“Introducción al mapeo de ejemplos”, Matt Wynne, <https://cucumber.io/blog/introducción-a-mapeo-de-ejemplo/> “Estrategia

de los Tres Amigos”, George Dinwiddie, <https://www.agileconnection.com/article/tres-amigos-estrategia-desarrollo-historias-de-usuarios> “La primera sesión

de mobbing de nuestro equipo”, Lisi Hocke, <https://www.lisihocke.com/2017/04/nuestros-equipo-primer-sesión-de-mobbing.html>

“El conductor-navegador en una combinación de estilo fuerte”, Maaret Pyhäjärvi, <https://medium.com/@maaret.pyhajarvi/the-driver-navigator-in-strong-style-pairing-2df0ecb4f657>

Guía de programación de pares de estilo fuerte y programación de mafias, Maaret Pyhäjärvi, <https://leanpub.com/u/maaretp>

Prueba exploratoria

Explórelo: reduzca el riesgo y aumente la confianza con pruebas exploratorias, Elisabeth Hendrickson, 2013 , <https://pragprog.com/book/ehxta/explore-it>

Pruebas exploratorias, Maaret Pyhäjärvi, <https://leanpub.com/exploratorytesting>

DevOps, Monitoreo, Observabilidad

Una guía práctica para realizar pruebas en DevOps, Katrina Clokie, <https://leanpub.com/testingindevops>

“Pruebas en producción de forma segura”, Cindy Sridharan, <https://medium.com/@copyconstruct/testing-in-production-the-safe-way-18ca102d0ef1>

"Monitoreo y observabilidad", Cindy Sridharan, <https://medium.com/@copyconstruct/monitoreo-y-observabilidad-8417d1952e1c>

"Charity Majors sobre la observabilidad y la comprensión de las ramificaciones operativas de un sistema", entrevista de InfoQ con Charity Mayores, <https://www.infoq.com/articles/charity-majors-observability-failience>

Ingeniería de confiabilidad del sitio de Google, <https://landing.google.com/sre/>

"¿Qué es la Ingeniería del Caos² ?" Joe Colontonio (incluye enlace a podcast con Tammy Butow), <https://www.joecolantonio.com/chaos-engineering/>

"Seguimiento, registro y seguimiento: ¿cuál es la diferencia?" crissy Kidd, <https://www.bmc.com/blogs/monitoring-logging-tracing/>

Automatización de pruebas

"Mantenga sus pruebas automatizadas simples y evite los antipatrones", Lisa Crispin, <https://www.mabl.com/blog/keep-your-automated-testing-simple>

"Automatización de pruebas: cinco preguntas que conducen a cinco heurísticas", Joep Shuurkes, <https://testingcurve.wordpress.com/2015/03/24/test-automation-five-questions-leading-to-five-heuristics/>

"Prácticas poderosas de automatización de pruebas", partes 1 y 2, Lisa Crispin y Steve Vance, <https://www.mabl.com/blog/powerful-test-automation-practices-pt-1> , <https://www.mabl.com/blog/powerful-test-automation-practices-pt-2>

"Diseño de conjunto de pruebas", Ashley Hunsberger, <https://github.com/ahunsberger/pruebaSuiteDiseño>

Acelerar: la ciencia de Lean y DevOps, Nicole Forsgren, et al, <https://itrevolution.com/book/accelerate/>

² <https://www.joecolantonio.com/chaos-engineering/>

"Análisis de fallas de pruebas automatizadas", Lisa Crispin, <https://www.mabl.com/blog/lisa-webinar-analyzing-automated-ui-test-failures>

"El iceberg de prueba", Seb Rose, <http://claysnow.co.uk/the-testing-iceberg/>

"¿Automatización y pruebas de nivel inferior? ¡Sea más preciso! ¡El triángulo de la automatización revisado nuevamente! Toyer Mamoojee, <https://toyerm.wordpress.com/2018/10/16/lower-level-automation-and-testing-be-more-precise-the-automation-triangle-revisited-again>

La guía del equipo para la capacidad de prueba del software, Ash Winter y Rob Meaney, <https://leanpub.com/softwaretestability>

Sobre los autores

Janet Gregory es entrenadora de pruebas ágiles y consultora de procesos en DragonFire Inc. Sus pares la votaron como la persona profesional de pruebas ágiles más influyente en 2015. Janet se especializa en mostrar a los equipos ágiles cómo las prácticas de prueba son necesarias para desarrollar productos de buena calidad. Imparte cursos de pruebas ágiles en todo el mundo y le gusta pasar los veranos en las montañas a las afueras de Calgary.

Lisa Crispin ha estado difundiendo la alegría ágil en el mundo de las pruebas y probando la alegría en el mundo ágil durante dos décadas y sus compañeros la votaron como la persona profesional de pruebas ágiles más influyente en 2012. Sus intereses actuales incluyen ayudar a los equipos a tener éxito con la entrega e implementación continuas y es un defensor de las pruebas que trabaja en mabl para explorar prácticas líderes en pruebas en la comunidad de software. Lisa vive con su marido, burros, gatos y perros en el hermoso Vermont.

Janet y Lisa son autoras de Pruebas ágiles condensadas: una breve introducción (2019), Pruebas más ágiles: viajes de aprendizaje para todo el equipo (2014), Pruebas ágiles: una guía práctica para evaluadores y equipos ágiles (2009), LiveLessons Agile Testing Curso en vídeo Essentials y curso de capacitación de 3 días “El enfoque de todo el equipo para las pruebas ágiles” ofrecido a través de Agile Testing Fellowship.

Juntos, fundaron la beca para hacer crecer una comunidad de profesionales que se preocupan por la calidad.

Janet Gregory: janetgregory.ca² Gorjeo: @janetgregoryca

Lisa Crispin: lisacrispin.com² Twitter: @lisacrispin Beca

de pruebas ágiles: agiletestingfellow.com² agiletester.ca²

² <https://janetgregory.ca/>

² <https://lisacrispin.com/>

² <https://agiletestingfellow.com/>

² <https://agiletester.ca/>