REICIS. Revista Española de Innovación, Calidad e Ingeniería del Software
E-ISSN: 1885-4486
reicis@ati.es
Asociación de Técnicos de Informática
España

Yagüe, Agustin; Garbajosa, Juan
Comparativa práctica de las pruebas en entornos tradicionales y ágiles
REICIS. Revista Española de Innovación, Calidad e Ingeniería del Software, vol. 5, núm. 4, diciembre-enero, 2009, pp. 19-32
Asociación de Técnicos de Informática
Madrid, España

Disponible en: http://www.redalyc.org/articulo.oa?id=92217159004

- Cómo citar el artículo
- Número completo
- Más información del artículo
- Página de la revista en redalyc.org

Sistema de Información Científica
Red de Revistas Científicas de América Latina, el Caribe, España y Portugal
Proyecto académico sin fines de lucro, desarrollado bajo la iniciativa de acceso abierto

Comparativa práctica de las pruebas en entornos tradicionales y ágiles

Agustin Yagüe y Juan Garbajosa
System and Software Technology Group (SYST)
E.U. Informatica, Univ.Politecnica de Madrid (UPM), Ctra. Valencia Km. 7, Madrid 28031
ayague@eui.upm.es, jgs@eui.upm.es

Resumen

Las pruebas han adquirido relevancia en el desarrollo de software. Sin embargo cómo y cuándo se aplican las técnicas de pruebas puede ser diferente dependiendo de la comunidad que las use, incluso aunque en ambas se usen las mismas técnicas. Para algunas comunidades las pruebas del software son un proceso en sí mismo, mientras que para otras es un actividad o una tarea más dentro del proceso de verificación y validación. Por otro lado, las metodologías ágiles están cambiando el paisaje del desarrollo. Cuando se aplican metodologías ágiles, se escribe código para superar las pruebas que, previamente, se han especificado. En este entorno, las pruebas pueden sustituir a la especificación de requisitos. Por lo tanto, los conceptos que subyacen a las pruebas son diferentes en ambos enfoques. En esta contribución se analizan las perspectivas convencionales y ágiles y se presentan algunas implicaciones desde el punto de vista de la ingeniería del software.

Palabras clave: Técnicas de pruebas, semántica de pruebas, metodologías convencionales, metodologías tradicionales, metodologías agiles.

Comparison in practice of software testing in conventional and agile approaches

Abstract

The relevance of tests in software development has grown up.. Nevertheless, how and when testing techniques are applied could be very different depending on the development community, even when different development communities use the same techniques. Software testing is a process in some communities, but sometimes, it is an activity or a task of the verification and validation process. Also, agile methodologies are changing the trend in software development. In agile methodologies, the source code of a program is written to pass a set of tests that have been defined in advance. In this scenario, tests are being used to substitute the software requirements specification as well. Therefore, the concepts underlying software testing are different in conventional and agile approaches. This contribution analyzes software testing from conventional and agile approaches and we present some findings from the software engineering perspective.

Keywords: Software testing techniques, testing approaches, conventional methodologies, traditional methodologies, agile methodologies.

Yagüe A., Garbajosa, J.,"Comparativa práctica de las pruebas en entornos tradicionales y ágiles", REICIS, vol. 5, no.4, 2009, pp.19-18.
Recibido: 23-7-2009; revisado: 28-10-2009; aceptado: 19-11-2009

1. Introducción

El término "prueba" se ha utilizado a lo largo de los años para referenciar diferentes conceptos: haciendo mención a técnicas para realizar pruebas (pruebas de caja blanca y de caja negra), dando nombre a diferentes actividades y objetivos en la forma de aplicar las pruebas (unitarias, integración, aceptación o sistema), presentando diferentes metodologías de desarrollo de software centrados en la realización de las pruebas (TDD – Test Driven Development, ATDD – Acceptance Test Driven Development, STDD – Story Test Driven Development) [20-21] o, incluso para dar soporte a nuevas metodologías relacionadas con los proyectos que tienen como objetivo realizar pruebas como TMAP [1].

Por otra parte, los organismos internacionales de normalización han documentado desde diferentes puntos de vista y en forma de múltiples estándares, las prácticas relacionadas con las pruebas; algunos de los cuales son [2-6]. En SWEBOK [7] las pruebas se presentan como una actividad que se desarrolla para evaluar la calidad de un producto y mejorarlo mediante la identificación de los defectos y los problemas. Este preámbulo muestra que las pruebas han sido ampliamente estudiadas y analizadas desde perspectivas muy dispares. De hecho, las pruebas son utilizadas por todas las comunidades de desarrollo de software y sistemas. Aun cuando las técnicas y enfoques son compartidas por las diferentes comunidades, es bastante corriente que las apliquen en diferentes fases del proceso de desarrollo, incluso en ámbitos distintos y mediante actores diferentes.

Las consideradas como metodologías convencionales consideran la ejecución de las pruebas como una actividad que se lleva a cabo una vez terminada la fase de codificación y que tiene como propósito la identificación de fallos tal como se describe en [2][3][4][5] y [17]. Esto no está en contradicción con que las pruebas puedan empezar a diseñarse desde las fases tempranas del proceso de desarrollo. Este entendimiento ha ido evolucionando y en la actualidad, las pruebas se consideran como una actividad integrada en todo el proceso de desarrollo.

Las metodologías ágiles han emergido como una reacción para superar algunos retos que la industria del software había identificado. Entre estos se encuentran los, impredecibles a veces, cambios en el mercado y una progresiva reducción del tiempo requerido para la comercialización [8]. En este sentido, las metodologías ágiles intentan

incrementar la calidad del producto y reducir el coste que se deriva de los cambios en los requisitos. Este objetivo se alcanza mediante la simplificación de los procesos relativos a los requisitos y las tareas de documentación. Para conseguirlo, se definen un conjunto de valores, principios y prácticas que, tomando las pruebas como punto central, promueven una rápida, flexible y continua comunicación entre los clientes y el equipo de desarrollo [9, 10].

En resumen, se puede considerar que las pruebas alcanzan objetivos diferentes: en las metodologías convencionales se utilizan principalmente como base de la verificación y la validación del producto desarrollado, mientras que en las metodologías ágiles se llegan a utilizar en sustitución de las especificaciones de requisitos y como guía para el desarrollo de software. Esto nos permite afirmar que las pruebas del software han ido evolucionado para poder desempeñar nuevos papeles en el desarrollo de software y sistemas.

En este artículo se analizan las pruebas del software desde ambas perspectivas, convencionales y ágiles con el objetivo de comparar el papel que desempeñan las pruebas en las metodologías ágiles y las convencionales, identificar similitudes y diferencias tanto en las técnicas, como en las estrategias y enfoques. Se muestra cómo pueden tener objetivos claramente diferentes. Este análisis se ha realizado en base a la información de los datos prácticos recogidos en múltiples encuestas en diferentes paises y en diferentes sectores. Las encuestas aparecen referenciadas al final del documento, además se ha realizado una encuesta a nivel del estado español1 sobre el nivel de implantación de las pruebas y que se encuentra en proceso de análisis.

El resto del artículo está organizado como sigue: La sección 2 presentará las técnicas básicas de pruebas. La sección 3 presentan las pruebas dentro de los enfoques convencionales. La sección 4 muestra las pruebas desde el punto de vista de las metodologías ágiles. Finalmente, la sección 5 presenta las principales conclusiones del artículo.

2. Técnicas básicas de pruebas

En esta sección se describen brevemente las técnicas de pruebas y su aplicación en ambos enfoques, convencional y ágil. Se han considerado dos categorías: caja blanca y caja negra

[1] La encuesta se realizó durante el 1er Agile Open Spain celebrado en Madrid los días 23 y 24 de octubre de 2009.

[7]. Por un lado, las técnicas de caja blanca (caminos básicos, control de flujo, control de datos o pruebas de ramificación) están basadas en estudiar el código fuente y se utilizan, principalmente, desde una perspectiva interna en el desarrollo de software. Como están basadas en el estado actual del código fuente, si se realizan cambios en la implementación, en la mayoría de los casos, también habrá que realizar cambios en los casos de pruebas. Este tipo de pruebas requieren una alta cualificación en el equipo de pruebas (o en su caso del de desarrollo), tanto para la identificación de los casos de pruebas como para su implementación. Incluso, aunque las pruebas de caja blanca se pueden aplicar en diferentes actividades de pruebas (unitarias, integración y sistema) fundamentalmente se aplican en el ámbito de las pruebas unitarias y a través de herramientas de ejecución automática de pruebas como por ejemplo Clover[2] o Cobertura[3] que permiten conocer la cobertura de código de las pruebas diseñadas.

No se han encontrado estudios, tanto nacionales como internacionales, que analicen en detalle la adopción práctica en los proyectos de desarrollo de cada una de las técnicas de caja blanca. En oposición a las pruebas de caja blanca, se encuentran las pruebas de caja negra (particiones de equivalencia, análisis de valores límite, pruebas transversales, etc.) tienen una visión externa del producto software y no están centradas en el código fuente. Estas pruebas están centradas en analizar la funcionalidad. Por lo tanto los casos de prueba se basan en las diferentes entradas que puede recibir el software y sus correspondientes valores de salida. Estas técnicas se pueden aplicar a cualquiera de los niveles de pruebas (unitarias, integración, aceptación) con diferentes niveles de abstracción en la definición de los casos de prueba.

Un aspecto interesante a resaltar es el perfil de los actores que llevan a cabo las pruebas del software en las metodologías ágiles y en las convencionales. En las metodologías ágiles son los ingenieros de desarrollo los encargados de ejecutar este tipo de pruebas mediante pruebas unitarias. Como estos ingenieros están trabajando a nivel de código fuente, conocen la estructura del software y, por lo tanto, pueden definir sin un mayor esfuerzo los casos de prueba adecuados para probar el código. En el caso de que se produzca un fallo, conocen las líneas de código que se ven afectadas por el caso de prueba

[2] http://www.atlassian.com/software/clover/ - Visitado en junio de 2009
[3] http://cobertura.sourceforge.net/ - Visitado en junio de 2009

que ha fallado y que ha revelado el problema. En las metodologías convencionales, puede ocurrir que el equipo de desarrollo defina y ejecute ciertas pruebas, pero son los ingenieros de pruebas los que definen los casos de pruebas mayoritariamente y el equipo de pruebas es el responsable de ejecutarlo.

Fundamentalmente, en las metodologías convencionales la pruebas de caja negra las lleva a cabo el equipo de pruebas o equipos de pruebas independientes4. Sin embargo, en las metodologías ágiles se presentan dos perspectivas: por un lado, los ingenieros de desarrollo que ejecutan las pruebas de caja negra a nivel de pruebas unitarias, puesto que conocen los tipos de datos de cada uno de los parámetros. Por otro lado, el resto del equipo, que aunque no está directamente involucrado en el desarrollo, también participa en el diseño y ejecución de las pruebas [15]. Aunque es cierto que estos aspectos pueden depender de los equipos de trabajo. No se puede considerar que existan unas técnicas que se utilicen más en unas metodologías que en otras y, por lo tanto, no se puede afirmar que las técnicas básicas tengan un papel diferente en las metodologías convencionales y en ágiles. En ambas metodologías, es muy importante la existencia de herramientas que, de forma automática, tomen medidas sobre la calidad del código o sobre el grado de cobertura que alcanzan los test diseñados (ya sea sobre pruebas de caja blanca o pruebas de caja negra).

Con respecto a las técnicas de pruebas, puede deducirse del estudio realizado por Fernández [29] sobre la aplicación de pruebas del software (aunque no se mencionan este tipo de pruebas de forma explícita) que existe un largo camino por recorrer en la planificación y el análisis de la cobertura de las pruebas, ya sean de caja blanca o de caja negra. Más del 50% de los encuestados diseñaban las pruebas justo antes de ejecutarlas o cuando ya estaba disponible el código.

3 Las pruebas en los enfoques convencionales

Analizando las pruebas desde el punto de vista de los enfoques convencionales, fundamentalmente se identifican cuatro actividades relacionadas: pruebas unitarias, pruebas de integración, pruebas de aceptación y pruebas de sistema [7]. En estas metodologías se presta especial atención a la elaboración de la especificación de requisitos software. Dicha especificación se refina dentro del proceso de identificación de los requisitos del software

[4] http://www.cdainfo.com/Down/5-Test/Status.pdf - Visitado en noviembre 2009

mediante iteraciones. En este enfoque, el proceso de desarrollo se dirige desde la especificación de requisitos hacia el código y posteriormente desde el código hasta las pruebas.

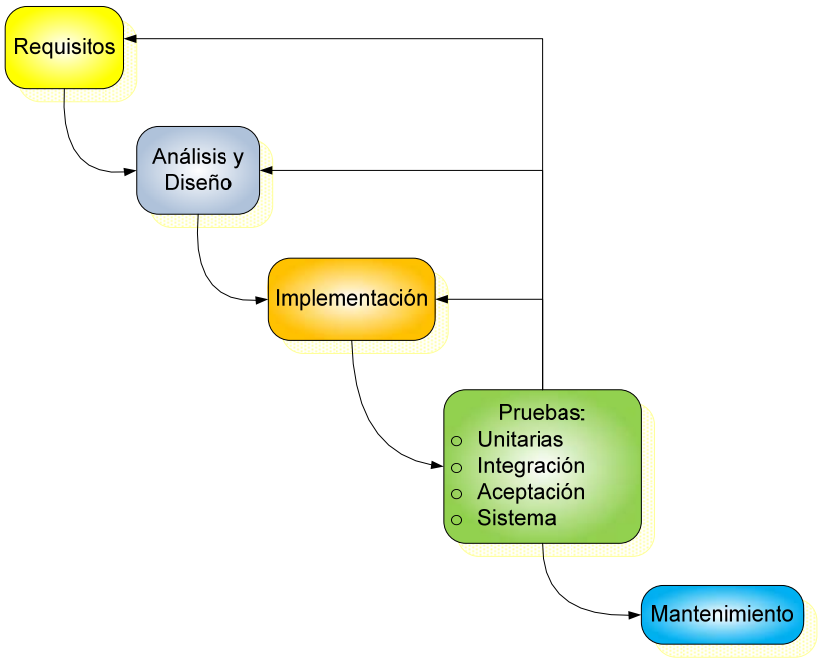

Figura 1. Enfoque tradicional de prueba.

En consecuencia, aunque en las metodologías convencionales las pruebas son importantes, éstas dependen directamente del resto de actividades del proceso de desarrollo. En [29] se comprueba que casi en el 60% de los proyectos, las pruebas se diseñan cuando ya está el código o cuando se comienza la fase de pruebas. Se puede considerar que en estos casos, el proceso de validación se entiende como un complemento del proceso de desarrollo.

La Figura. 1 muestra una visión general del modelo convencional de desarrollo donde todas las actividades se han aplicado en secuencia; esto quiere decir que las pruebas de integración no se pueden llevar a cabo hasta que no se han terminado todas la pruebas unitarias, lo que es extensible al resto de actividades. Por supuesto, este es un caso extremo de secuencialidad. Por otro lado, en [16] se puede encontrar un ejemplo de aplicación donde el proceso de pruebas no puede comenzar hasta que el proceso anterior en el tiempo no haya finalizado. Eso no quiere decir que la especificación de los procedimientos de pruebas no haya comenzado antes en el ciclo de vida. Mientras tanto, en el modelo en V

[17] y también en [7], la definición de los casos de prueba comienza en las primeras fases del modelo de procesos: las pruebas de aceptación y las pruebas de sistema se derivan del documento de especificación de requisitos, las pruebas de integración se derivan de la documentación del diseño y las pruebas unitarias de las de implementación de los módulos. Dentro de los enfoque convencionales y según se menciona en los resultados del estudio sobre las pruebas del software5 elaborado por el Quality Assurance Institute entre los asistentes a la International Conference on Software Testing, más del 70% de los proyectos desarrollados, disponen de procesos de pruebas documentados. Además, alrededor del 80% de los proyectos disponen de planes de pruebas detallados, casos de prueba, etc. Asimismo, en dicho informe se refleja que los resultados de las pruebas se utilizan para informar de los errores detectados y de los porcentajes de pruebas ejecutadas satisfactoriamente. Resulta bastante evidente que podemos esperar que justamente los asistentes a este congreso estén muy concienciados con el papel de las pruebas, por lo que estos resultados no son extrapolables a otras comunidades.

Como resumen, el enfoque de las pruebas en las metodologías convencionales se basa en la definición de los casos de prueba en función de unos requisitos previamente establecidos, mediante los lenguajes adecuados, por ejmeplo XUnit, FIT[17] o similares. Por lo tanto, el objetivo principal es comprobar que el desarrollo realizado cumple los requisitos definidos.

4. Las pruebas en las metodologías ágiles

Las metodologías ágiles en general, y particularmente Extreme Programming (XP) [19], son de alguna manera los responsables del aumento de popularidad de las pruebas del software. En XP las necesidades de los usuarios no se representan mediante documentos estandarizados de requisitos, sino que se representan mediante unos artefactos denominados "historias de usuario" que representan de una forma particular los requisitos del sistema. Cada historia de usuario, entre otras cosas, lleva asociada una lista de criterios de aceptación y, para cada criterio de aceptación, se define el conjunto de casos de prueba necesarios para validar el criterio de aceptación. Estas tareas de definición se realizan en las fases tempranas del proyecto y antes de que se comience la implementación del sistema.

[5] http://www.cdainfo.com/Down/5-Test/Status.pdf - Visitado en noviembre 2009

Con esto se consigue que las pruebas se utilicen para validar las necesidades de los usuarios y para dirigir la implementación. Se puede considerar que las historias de usuario y, por extensión, las pruebas asociadas a cada una juegan el papel de especificaciones del sistema.

En los enfoque ágiles las pruebas son el centro de la metodología y, por lo tanto son ellas las dirigen el proceso de desarrollo. Las metodologías ágiles plantean que el desarrollo no es un conjunto de fases en las que las pruebas son una fase más, sino que abogan porque las prácticas y el desarrollo estén completamente integradas, lo que puede llevar a modificar las estructuras organizativas de las empresas [15]. La irrupción de las metodologías ágiles, como se puede comprobar en [23-24], ha llevado a que cerca del 70% de las empresas ya estén incorporando algunas prácticas ágiles en su proceso de desarrollo. Según estos autores, esto ha traído como consecuencia una mejora de la calidad de los productos entregados en casi un 70% de los proyectos, como puede verse en la Figura. 2.

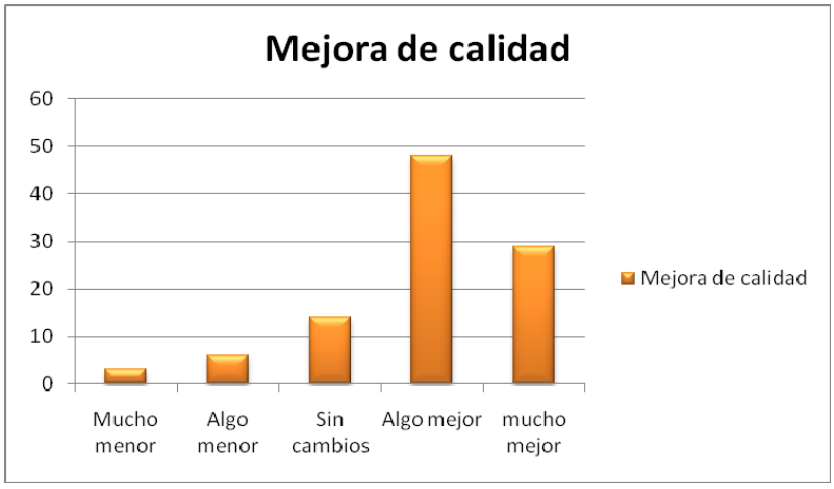

Figura. 2 Impacto de las metodologías ágiles en el aumento de calidad. Basado en [23]

Además de todo lo mencionado, es importante destacar que las metodologías ágiles no consideran las pruebas como un conjunto de niveles que haya que ir superando para alcanzar la validación final del sistema que se está desarrollando. Las metodologías ágiles presentan distintos enfoques del proceso de desarrollo que vienen determinados por los tipos de pruebas que se realizan.

En este artículo se van a considerar dos metodologías TDD [22] y ATDD [20]. Ambas metodologías, según se muestra en [25] son las más demandadas y aquellas que las

personas relacionadas con desarrollo ágil quieren adoptar (cerca del 50% de las personas encuestadas y que no estaban todavía aplicando este tipo de metodologías).

TDD y ATDD comparten el punto de vista desde el punto de vista del proceso de desarrollo pero utilizan diferentes herramientas de trabajo: las pruebas unitarias en TDD y las pruebas de aceptación en ATDD.

En el caso de TDD, cada requisito se representa como un artefacto denominado historia de usuario al que se le asocian sus correspondientes pruebas unitarias. Cuando se aplica TDD, en primer lugar define el repositorio de historias de usuario, conocido como "*Product backlog*", que representa los requisitos del sistema. Posteriormente, los desarrolladores escriben las pruebas unitarias necesarias para satisfacer cada una de las historias de usuario y, una vez escritas las pruebas, comienzan a implementar el código fuente necesario para superarlas. Por lo tanto, la lista completa de las historias de usuario y sus casos de prueba adoptan el mismo papel que la especificación de requisitos software en las metodologías convencionales y, las pruebas, son las que guían el desarrollo. Existen trabajos empíricos sobre el papel de las pruebas, en [28] se muestra que se empiezan a considerar las pruebas como mecanismo de especificación de requisitos (45%), sólo superados por documentos de texto (52%). Además, cuando se trata de la captura de especificaciones de diseño, se convierte en la práctica más utilizada (57%). El papel de las pruebas, como sustitutivo de los requisitos convencionales y según se presenta en [11-15], nos permite subsanar algunos problemas que tienen las metodologías ágiles en relación con la identificación de requisitos.

Un aspecto importante que provoca confusión es que no es suficiente escribir pruebas unitarias para que pueda considerar que se está aplicando TDD. Las pruebas unitarias podrían limitarse a probar una parte del código aislada del resto, mientras que TDD representa el proceso de desarrollo y tiene como objetivo comprobar que la aplicación funcionará correctamente. Por lo tanto en TDD las pruebas unitarias guían el proceso de desarrollo. Cuando se definen qué pruebas vamos a realizar en TDD no se tienen en cuenta decisiones de diseño mientras que cuando se escribe cada prueba unitaria, hay que tener en cuenta lo que hace el código implementado.

ATDD se basa en las pruebas de aceptación. Este enfoque toma como punto de referencia al usuario. El proceso es semejante a TDD pero guiado por las pruebas de

aceptación. Como se presenta en [28], TDD es el enfoque más aplicado para validación y pruebas, siendo su aplicación casi el doble que ATDD (71% y 40%).

Cuando se aumenta el nivel de abstracción y se aplica ATDD, cada historia de usuario tiene asociados un conjunto de criterios de aceptación. Posteriormente, para cada criterio de aceptación se definen las pruebas de aceptación que se deben superar. Una vez que las pruebas de aceptación están claramente definidas, el equipo de desarrollo comienza a escribir el código fuente que es necesario para superar los criterios de aceptación. Durante el proceso de desarrollo también será necesaria la definición de pruebas unitarias, asociadas al código que se implementa que garanticen que el código cumple con los requisitos. Como ocurre en TDD, la lista completa de historias de usuario (*product backlog*) junto con sus criterios de aceptación y las pruebas de aceptación desempeñan un papel equivalente a las especificaciones de requisitos software en las metodologías convencionales.

Como en el caso de TDD, las pruebas de aceptación no son ATDD. Las pruebas de aceptación tienen como objetivo probar la funcionalidad del sistema mientras que ATDD representa el proceso de desarrollo y tiene como objetivo que el sistema se construya de acuerdo a la funcionalidad determinada por el usuario. ATDD se lleva a cabo en colaboración con el usuario y en un lenguaje que usuario pueda entender. Para ello, las pruebas de aceptación se realizan en un lenguaje que sea entendible por parte del usuario; sirvan como ejemplo FIT o Easyaccept [18, 21].

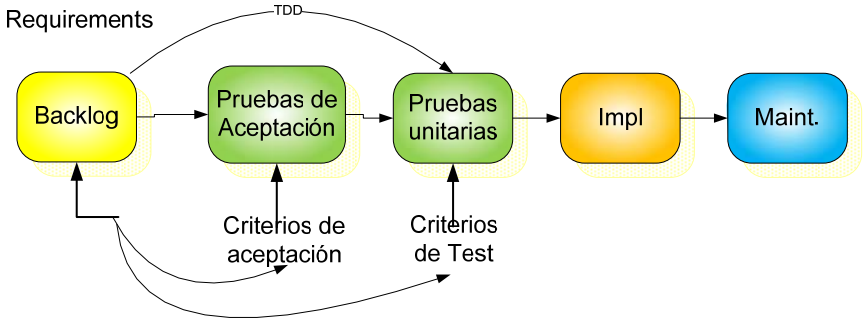

Figura. 3. Enfoque de ágil de pruebas.

La Figura 3 muestra la principal estrategia seguida en las metodologías ágiles. En este caso, las metodologías ágiles utilizan las pruebas unitarias y de aceptación como principales herramientas para dar soporte a los enfoques de pruebas. Considerando los

hábitos y su aplicación metodológica, hace que tanto TDD como ATDD se encuentren como una de las prácticas más adecuadas en cada sesión [25].

Debe realizarse una mención aparte sobre las pruebas de integración y las pruebas de sistema. Ambas se aplican de forma implícita en el proceso de desarrollo, es decir, cada nueva parte del código se escribe y se integra con el sistema completo, por lo tanto, no es necesario escribir pruebas específicas de integración sino que éstas se encuentran incorporadas en el resto de pruebas que se han desarrollado.

Para poder dar soporte a estos enfoques de pruebas es necesario disponer de una mínima infraestructura que permita automatizar la ejecución del elevado número de pruebas que se diseñan y la toma de medidas que permitan conocer el nivel de calidad que está alcanzando el desarrollo. Esta infraestructura mínima recibe el nombre de entorno de integración continua. Los entornos de integración continua, si bien son aplicables tanto en enfoques convencionales como ágiles, han tomado especial relevancia en éstos últimos. Estos entornos se caracterizan por la integración de múltiples herramientas que dan soporte al respaldo automatizado del código, su compilación, la ejecución de las pruebas, la toma de medidas de calidad, la generación de documentación y el despliegue de la aplicación. Dado que el proceso de ejecución de las pruebas está automatizado, cada vez que se realiza la integración de una nueva funcionalidad al sistema desarrollado, se llevan a cabo pruebas de regresión que aseguren que no se ha introducido ningún error nuevo en el sistema.

5 Conclusión

Como ya se ha justificado previamente, las pruebas desempeñan un papel importante en los diferentes modelos de procesos que van desde los modelos convencionales a los ágiles. Este artículo ha discutido cómo el papel de las pruebas es diferente en las metodologías convencionales y ágiles. Mientras en las metodologías convencionales, las pruebas, formalmente hablando, se ejecutarán mayoritariamente una vez que el código esté terminado, aunque puedan diseñarse antes de la codificación, en las ágiles las pruebas se escriben antes comenzar la codificación y el código que se escribe es el exclusivamente necesario para superar las pruebas y guían el proceso de desarrollo. El hecho de que las pruebas se realicen tan pronto como sea posible, está alineado con la reducción del impacto del coste de la detección de errores en las fases de pruebas.

En las metodologías ágiles, no se distingue de forma específica las pruebas de integración sino que éstas simplemente se incorporan al conjunto de las pruebas que se definen ya sea en la aplicación de TDD o de ATDD.

Desde el punto de vista de las técnicas básicas de pruebas, ambas metodologías aplican las mismas técnicas básicas de pruebas: caja blanca y caja negra. Los actores y el papel que desempeñan las técnicas de pruebas son algo diferentes. Aunque las prácticas puedan parecer semejantes a nivel práctico, no lo son a nivel semántico.

En lo referente a automatización, para lograr la integración en cada una de las iteraciones, se aplican entornos de integración continua. Estos entornos además permiten la realización automática de las pruebas de regresión del sistema desarrollado.

Finalmente, este papel cambiante de las pruebas también tiene una fuerte implicación en la definición general del proceso de desarrollo y que debe ser tenido en cuenta por los estándares de desarrollo.

Agradecimientos

Este trabajo ha sido parcialmente financiado por el Ministerio de Educación y Ciencia a través del proyecto OVAL/PM y por el Ministerio de Ciencia y Tecnología en el marco del proyecto FLEXI FIT-340005-2007-37 (ITEA2 6022).

Referencias

[1] Koomen, T., van der Aalst, L., Broekman, B., Vroon, M.: *TMap Next, for result driven testing*. UTN Publishers, 2006.

[2] ISO: *ISO/IEC 12207. Software Engineering - Life Cycle Processes*. ISO, 2008.

[3] ISO: *ISO/IEC 15288. Systems Engineering - System Life Cycle Processes*. ISO, 2008.

[4] IEEE: *IEEE Std 1008, Standard for Software Unit Testing. IEEE*, pub-IEEE-STD, 2002.

[5] BSI: *BS 7925-2:1998 - Software testing. Software component testing*. BSI, 1998.

[6] European Consortium for Space Standardisation, E.: *Software - part 1: Principles and requirements*. E-40, 2003.

[7] Abran, A., Moore, J.W., Bourque, P., Dupuis, R., Tripp, L.L.: *Guide to the Software Engineering Body of Knowledge (SWEBOK)*. IEEE, 2004.

[8] Boehm, B.: *"A view of 20th and 21st century software engineering"*. *"Proceedings of the 28th Intl. Conf. on Software engineering*", ACM, pp. 12-29, 2006.

[9] Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., et al.: *Manifesto for agile software development*, http://agilemanifesto.org/, 2001.

[10] Shore, J., Warden, S., *The Art of Agile Development*. O'Reilly Media, 2007.

[11] Paetsch, F., Eberlein, A., Maurer, F., *"Requirements engineering and agile software development*". En *Proceedings of the Twelfth InternationalWorkshop on Enabling Technologies*, 2003.

[12] Cao, L. y Ramesh, B., *"Agile requirements engineering practices: An empirical study"*, IEEE Software, vol. 25, nº 1, pp. 60-67, 2008.

[13] Eberlein, A. y Leite, J., *"Agile requirements definition: A view from requirements engineering"*. En *Proceedings of the International Workshop on Time-Constrained Requirements Engineering (TCRE'02*), pp. 4-8, 2002,

[14] Dyba, T. y Dingsoyr, T.: *"Empirical studies of agile software development: A systematic review*", *Information and Software Technology*, vol.50, nº9-10, pp. 833-859, 2008.

[15] Talby, D., Hazzan, O., Dubinsky, Y. y Keren, A. *"Agile Software Testing in a Large-Scale Project"*, IEEE Software, vol. 23, nº 4, pp. 30-37, 2006.

[16] Royce, W.W., *"Managing the development of large software systems: concepts and techniques"*. En: *Proceedings of the 9th international conference on Software Engineering*, pp. 328-338, 1987.

[17] A. Bröhl y W. Dröschl, *Das V-Model*, Oldenbourg Verlag, 1995

[18] Cunningham, W., *FIT: Framework for integrated acceptance testing*. http://fit.c2.com, consultado en junio de 2009.

[19] Beck, K., *Extreme Programming Explained: Embrace Change*, Addison Wesley Professional, 2004.

[20] Koskela, L., *Test Driven: TDD and Acceptance TDD for Java Developers*, Manning Publications, 2007.

[21] Sauve, J.P., Neto, O.L.A. y Cirne, W., "Easyaccept: a tool to easily create, run and drive development with automated acceptance tests". En: *Proceedings of the 2006 international workshop on Automation of software test*, 2006, pp. 111-117

[22] Janzen, D. y Saiedian, H., "Test-driven development: Concepts, taxonomy, and future direction", Computer, vol. 38, nº 9, 2005, pp. 43-50.

[23] Ambler, S., *Agile Adoption Survey 2008*, www.ambysoft.com/surveys/agileFebruary2008.html, consultado en junio de 2009.

[24] Versionone, *The State of Agile Development, 3rd Annual Survey: 2008*. www.versionone.com/AgileSurvey/, consultado en junio de 2009.

[25] Ambler, S. y Vizdos, M., *Agile Practices Survey 2009*, www.ambysoft.com/surveys/practices2009.html, consultado en junio de 2009.

[26] Ambler, S. y Vizdos, M., *Agile Practices and Principles Survey 2008*, www.ambysoft.com/surveys/practicesPrinciples2008.html, consultado en junio de 2009.

[27] Ambler, S. y Vizdos, M., *Agile Project Initiation 2008*, www.ambysoft.com/surveys/projectInitiation2009.html, consultado en junio de 2009.

[28] Ambler, S., *Test-Driven Devlopment Survey 2008*, www.ambysoft.com/surveys/tdd2008.html, consultado en junio de 2009.

[29] Fernández, L. "Un sondeo sobre la práctica actual de pruebas de software en España", *REICIS, Revista Española de Innovación, Calidad e Ingeniería del Software*, vol. 1, nº 2, 2005, pp. 41-54.