

PRUEBAS DEL SISTEMA EN PROGRAMACIÓN EXTREMA

J. J. Gutiérrez, M. J. Escalona, M. Mejías, J. Torres

Department de Lenguajes y Sistemas Informáticos

University of Sevilla

{javierj, escalona, risoto, jtorres}@lsi.us.es

RESUMEN

Este trabajo analiza la integración de las pruebas del sistema, o pruebas funcionales, dentro de un desarrollo basado en eXtreme Programming. Este trabajo también estudia los problemas que presenta el desarrollo de este tipo de pruebas tomando como base los artefactos generados por XP y ofrecemos una solución a estos problemas mediante un proceso de generación de pruebas del sistema aplicable a los customer-in-situ.

PALABRAS CLAVES

Agile Methods, Extreme Programming, acceptance test, functional test, system testing, client in-situ.

1. INTRODUCCIÓN

En el primer punto se estudiará donde encajan las pruebas del sistema dentro de la clasificación de las pruebas de XP. En el segundo punto se definen con detalle las pruebas de aceptación según la definición de XP.

1.1. ¿Dónde encajan las pruebas del sistema en XP?

Uno de los pilares de la eXtreme Programming (XP a partir de ahora) es el proceso de pruebas [1]. XP anima a probar constantemente tanto como sea posible. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones.

XP divide las pruebas del sistema en dos grupos [1]: pruebas unitarias, encargadas de verificar el código y diseñada por los programadores, y pruebas de aceptación o pruebas funcionales destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida diseñadas por el cliente final.

Con solo dos opciones a elegir, ¿dónde encajan las pruebas funcionales o prueban del sistema?. Antes de responder a esta pregunta vamos a definir el concepto de prueba del sistema. Las pruebas del sistema tienen como objetivo verificar la funcionalidad del sistema a través de sus interfaces externas comprobando que dicha funcionalidad sea la esperada en función de los requisitos del sistema [13].

Generalmente las pruebas del sistema son desarrolladas por los programadores para verificar que su sistema se comporta de la manera esperada, por lo que podrían encajar dentro de la definición de pruebas unitarias que propone XP. Sin embargo, las pruebas del sistema tienen como objetivo verificar que el sistema cumple los requisitos establecidos por el usuario por lo que también pueden encajar dentro de la categoría de pruebas de aceptación.

En este artículo consideramos que las pruebas del sistema forman parte de las pruebas de aceptación, aunque no todas las pruebas de aceptación son pruebas del sistema. Por ejemplo, pruebas de usabilidad de la interfaz de usuario o pruebas de implantación del sistema en su entorno de producción entrarían en la categoría de pruebas de aceptación y no son pruebas del sistema.

1.2. Pruebas de aceptación en XP.

Acceptance Tests are an integral part of incremental development as practiced by XP. All User Stories are supported by Acceptance Tests, which are defined by the On-site Customer. These tests address the fears that the business has been misunderstood [2]. Las pruebas de aceptación son más importantes que las pruebas unitarias dado que significan la satisfacción del cliente con el producto desarrollado y el final de una iteración y el comienzo de la siguiente [3].

The Acceptance Tests force the customer to dig deep into their domain knowledge and precisely state what the application should do in specific circumstances [2], por esto, el cliente es la persona adecuada para diseñar las pruebas de aceptación [7]. Sin embargo esto supone el grave problema de que el cliente no tiene que tener, y en general no tiene, la formación adecuada para desarrollar buenas pruebas de aceptación. Por ejemplo, el cliente, en la mayoría de los casos sabe que es lo que quiere que la aplicación haga correctamente, pero puede no ser capaz de desarrollar un conjunto de pruebas que garantice la total cobertura de la funcionalidad especificada en la historia de uso, limitándose a probar que el sistema hace lo que debe sin verificar todas las variantes que pueden aparecer.

Existen en la actualidad muchas herramientas, como [10] y [11], para desarrollar pruebas de aceptación, lo suficientemente sencillas para que un cliente pueda manejarlas. Sin embargo estas herramientas son inútiles si el cliente es incapaz de diseñar un conjunto completo de pruebas que verifiquen toda la funcionalidad del sistema y no solo una parte o con unos valores concretos.

En este trabajo abordamos la necesidad de que los clientes cuenten con una guía metodológica para el desarrollo de pruebas de aceptación que cubran, si no todo, si la mayor cantidad posible de la funcionalidad registrada en una historia de uso, realizamos una propuesta para guiar el diseño de estas pruebas y mostramos un ejemplo de su aplicación basado en un caso real. En el apartado 2 profundizamos en el problema de generar pruebas del sistema en un desarrollo XP. En el apartado 3 mostramos un mal ejemplo que ilustra la necesidad de esta guía. En el apartado 4 proponemos una guía y, en el apartado 5 mostramos un ejemplo de su aplicación. Por último en el punto 6 recogemos las conclusiones y futuras investigaciones.

2. EL PROBLEMA DE LAS PRUEBAS DEL SISTEMA EN XP

Como hemos visto en el punto 1.1 el objetivo de las pruebas del sistema es verificar los requisitos. Por este motivo, los propios requisitos del sistema son la principal fuente de información a la hora de construir pruebas del sistema. En procesos de desarrollo no extremos, como el Proceso Unificado [17], la funcionalidad del sistema se recoge en los requisitos funcionales, expresados principalmente como casos de uso [15]. Estos casos de uso son completados con plantillas de texto que describen la secuencia principal o escenario de éxito, secuencias alternativas, tratamiento a los posibles errores, precondiciones, postcondiciones e invariantes. Además, el uso de plantillas permite recoger información adicional, si se estima conveniente, como la frecuencia de uso o la prioridad del caso de uso.

Toda esta información facilita la generación de casos de prueba. Conociendo el camino principal, los caminos alternativos, errores posibles y su tratamiento y precondiciones, postcondiciones e invariantes es posible explorar exhaustivamente todas las combinaciones posibles asegurando que, en todo momento, el sistema responde de acuerdo a sus requisitos. Diversos trabajos, como [14], muestran como es posible automatizar en parte este proceso.

Sin embargo, toda esa información sobre los requisitos del sistema no está disponible en XP. En un desarrollo XP, los requisitos, es decir lo que el sistema debe hacer, se recogen en story card durante el juego de la planificación. Una story card solo recoge una descripción en lenguaje natural, en general ambigua, incompleta e informal, de un fragmento de la funcionalidad del sistema. Aunque las story card pueden ser completadas con notas, el tamaño recomendado para la tarjeta no permite añadir información clara ni exhaustiva. Además, por estos motivos, es imposible automatizar la generación de pruebas del

sistema, siendo posible solo la automatización de su ejecución con las herramientas adecuadas. El proceso del juego de la planificación se describe con más detalle en el punto 4.1.

La solución a este problema es la misma solución adoptada a la hora de implementar los requisitos. Debido a la escasez de información sobre el comportamiento esperado del sistema se hace imprescindible contar con la colaboración de customer-in-situ para el desarrollo de las pruebas del sistema. Existen trabajos que proponen que el customer-in-situ desarrolle las pruebas del sistema mediante una herramienta captura y reproducción, en paralelo al desarrollo de la iteración [12 (creo)].

No conocemos estudios empíricos que midan la calidad de las pruebas desarrolladas por los customer-in-situ. Sin embargo, dado que con toda seguridad no tienen conocimientos de ingeniería de software ni de prueba del software, no creemos que sus suites de pruebas tengan la cobertura adecuada. Algunas técnicas clásicas, como el análisis de posibles caminos, valores límite o categorías equivalentes [16] son imprescindibles para garantizar que se verifica adecuadamente la implementación de los requisitos.

La solución que proponemos, y que describimos en detalle en las siguientes secciones, es el desarrollo de una metodología de diseño de pruebas del sistema lo suficiente simple para ser desarrollada por los customer-in-situ.

3. LA IMPORTANCIA DEL CLIENTE EN EL DISEÑO DE PRUEBAS DE ACEPTACIÓN.

Customers often have as much trouble writing functional tests as developers have writing unit tests [3]. Para ilustrar esto mostramos a continuación un ejemplo obtenido de un sistema real descrito en [9]. La historia de uso está recogida en la tabla 1.

Table 1. Autenticación del médico.

Quando un médico solicita entrar en el sistema, el sistema solicita su identificador y contraseña. Si el médico está registrado en el sistema accede a la pantalla de gestión de pacientes (RF-02). Si el identificador o la contraseña son incorrectos, el sistema vuelve a solicitarlos.

Existen propuestas específicas para XP para guiar el proceso de generación de pruebas de aceptación. En [5] se propone una cuyos puntos principales se recogen en la tabla 2.

Table 2. Pasos para escribir pruebas de aceptación.

1. Identify all the actions in the story
2. For each action, write two tests.
3. For somedata, supply inputs that should make the action succeed, and fill in whatever a successful result is for result.
4. For otherdata, supply inputs to make the action fail, and fill in what the response should be.

La propuesta [5] propone redactar tablas donde cada línea es una prueba con tres columnas: la acción a probar, los datos de prueba y el resultado esperado. Las pruebas resultado de aplicar esta propuesta se recogen en la tabla 3.

Table 3. Acceptance test template.

<i>Action</i>	<i>Data</i>	<i>Expected Result</i>
1. Login	(Somedata) ValidUser / ValidPassword	Success: access to pacient manager.
2.	(Otherdata) InvalidUser / InvalidPassword	Failure: login again

Sin embargo estas pruebas no cubren toda la funcionalidad que se puede extraer del caso de prueba. Algunas preguntas que quedan sin respuesta son: ¿Qué sucede cuando el nombre es válido y clave inválida?. ¿Obtenemos el mismo resultado que si el nombre es inválido y la clave válida o si ambos son inválidos?. ¿Cuántas veces se puede intentar acceder al sistema?. ¿Infinitas veces?.

Las respuestas a estas preguntas no están explícitamente recogidas en la historia de uso, por lo que es el cliente quien debe conocer la respuesta, o decidir cuales son las respuestas que desea en su sistema. Si el cliente no se había planteado estas cuestiones, puede ser un primer paso para identificar nueva funcionalidad. Por ejemplo se podría escribir una nueva historia que ofreciera la posibilidad de recordar la clave si el usuario la hubiera olvidado.

El problema es que una historia de uso no es una descripción completa y sistemática de la funcionalidad del sistema, sino solo un resumen de esta. Este resumen se completa con la información que el propio cliente conoce por ser experto en el dominio del problema. Las pruebas de aceptación no están basadas en las historias de uso sino en la funcionalidad que conoce el cliente asociada a dicha historia de uso. Por tanto cualquier propuesta de generación de pruebas basada solo en historias de uso será incompleta y, por eso, necesitamos que sea el usuario quien las escriba. Esta no es una idea nueva sino que aparece casi desde el inicio de la XP [12], sin embargo es difícil de llevar a la práctica por falta de experiencia de los clientes.

4. UNA PROPUESTA PARA DESARROLLO DE PRUEBAS DE ACEPTACIÓN.

The purpose of acceptance testing is for the users to verify the system or changes meet their original needs. The emphasis is on evaluating the system via normal business circumstances, but in a controlled testing environment.

Una de las misiones del cliente es escribir estas pruebas, realizarlas sobre el sistema y dar su aprobación al mismo.

4.1. Las pruebas de aceptación dentro de XP.

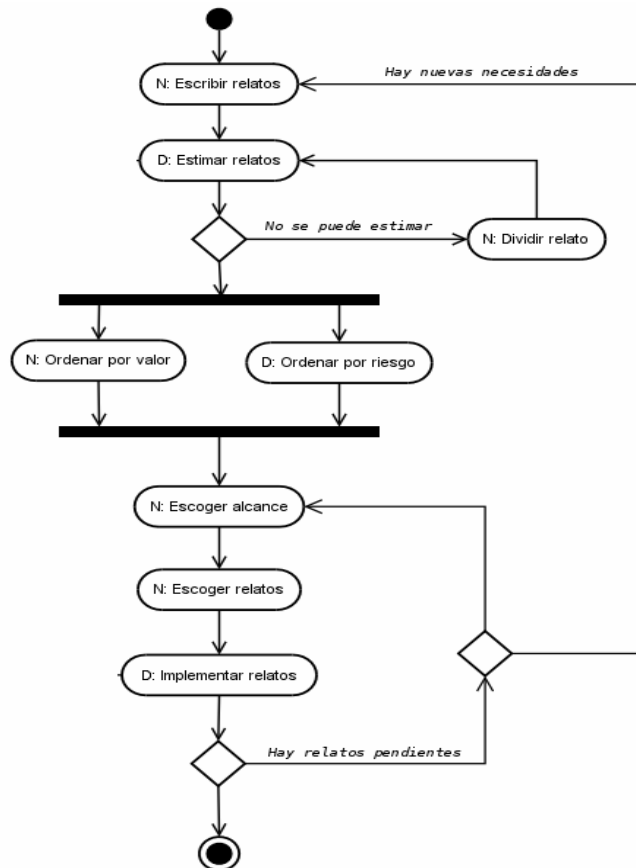
Un desarrollo mediante programación extrema está compuesto por una serie de iteraciones cortas. Cada iteración concluye ejecutando un conjunto de pruebas de aceptación que permitan al cliente comprobar si está satisfecho con el resultado.

En XP no existe una fase de requisitos propiamente dicha, en su lugar, al comienzo de cada iteración, se lleva a cabo el juego de la planificación [1] (ilustración 1).

There is no a requirement phase in XP. Instead of requirement phase, XP propose the planning game (Illustration 1).

Ilustración 1. Juego de la planificación.

N: Negocio
D: Desarrollo



En él, el cliente y el equipo de desarrollo negocian el alcance del proyecto para una iteración, identifican un conjunto de historias de usuario y seleccionan las más importantes para el cliente para ser implementadas en la siguiente iteración. Una historia de usuario es una descripción breve, de una o dos líneas, del comportamiento del sistema desde el punto de vista del usuario de dicho sistema [1].

Las pruebas de aceptación se elaboran a lo largo de la iteración, en paralelo con el desarrollo del sistema, y adaptándose a los cambios que el sistema sufra.

4.2. Descripción de la propuesta.

Una propuesta metodológica para guiar la generación de pruebas de aceptación por parte del cliente debe ser lo suficientemente sencilla para que cualquier persona sin experiencia en ingeniería del software y en pruebas pueda ponerla en práctica. Tampoco debe estar atada a ninguna notación o herramienta específica, aunque existen propuestas interesantes como las vistas en [6] y [10] consideramos interesante no atarse de antemano a ninguna herramienta.

Los pasos que proponemos en este trabajo para que el cliente pueda generar un conjunto de pruebas de aceptación que verifiquen completamente la funcionalidad de una historia de uso se resumen en la tabla 4 y en la ilustración 1. Esta propuesta se ha creado a partir de los puntos comunes identificados en [8] y mostrados en el apartado 3.

Table 4. Pasos para obtener un conjunto de pruebas de aceptación.

	<i>Descripción</i>	<i>Resultado</i>
1	Identificar todos los posibles resultados observables de la historia.	Listado de resultados observables.
2	Identificar los resultados que terminan la historia y los	Listado de resultados observables clasificados en

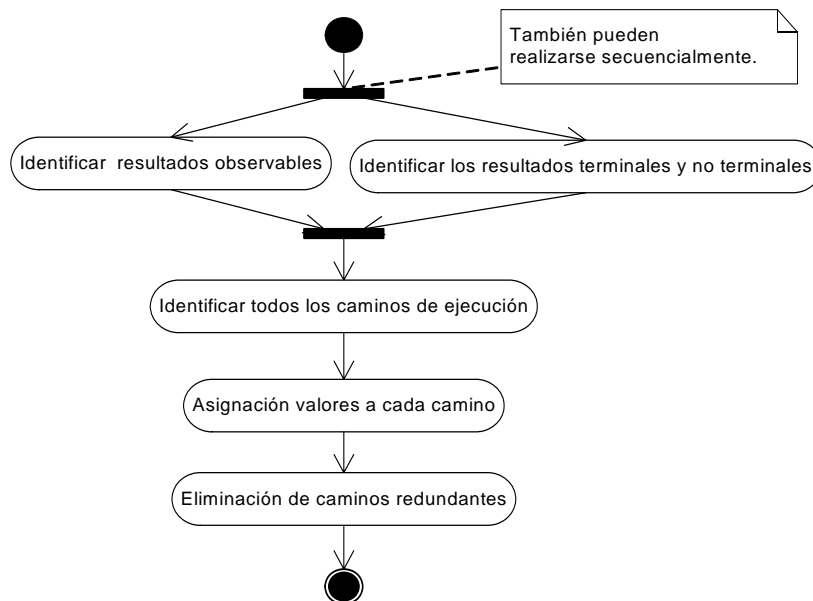
	que permiten continuar dentro la historia.	terminales y no terminales.
3	Identificar todos los caminos de ejecución posibles.	Listado de caminos de ejecución posibles y a cual de los resultados identificados conduce. List of execution paths and their result.
4	Asignar un conjunto de valores válidos y valores del entorno a cada camino de ejecución para obtener el resultado esperado. Assign a set of values and environment values to each path to obtain it result.	Listado de caminos de ejecución, con sus resultados esperados y los valores que permiten obtener dicho resultado. List of execution paths, with their results and the values needed.
5	Eliminación de caminos redundantes. Erase redundant paths.	Listado de caminos de ejecución, valores de prueba y resultados que se convertirán en pruebas de aceptación. List of execution paths that will be acceptance tests.

A continuación se describe con más detalle cada uno de los puntos y, en el siguiente apartado, se muestra un ejemplo de su aplicación.

En el primer punto, el cliente debe ser capaz de enumerar y describir brevemente que consecuencias va a tener su historia de uso. Un resultado observable puede ser una pantalla del sistema, un nuevo elemento almacenado / modificado o eliminado de una base de datos, un mensaje o petición que recibe otro ordenador o un servidor, etc... En resumen, algo que se pueda comprobar bien manualmente, bien mediante código.

Estos resultados pueden terminar o no la historia. Por ejemplo, en una historia de inserción de clientes podemos estar insertando clientes hasta que pulsemos la opción de salir del formulario de inserción. Al insertar un cliente el resultado observable puede ser un mensaje en el mismo formulario de inserción indicando que se ha insertado correctamente.

Ilustración 2. Diagrama de actividades para obtener pruebas de aceptación.



El segundo punto se puede realizar durante el paso anterior, o posteriormente una vez que se tiene la lista de todos los resultados observables. En el ejemplo de la inserción de clientes, un mensaje en el formulario de inserción indicando que el cliente se almacenó correctamente es un resultado observable que permite continuar con la historia (permite seguir insertando clientes), mientras que pulsar la opción

salir nos llevará a otro formulario (resultado observable) y pondrá fin a la historia (ya no podemos seguir insertando clientes).

Un camino de ejecución es una descripción de las interacciones entre un actor o actores y el sistema, dentro de la historia de uso, para llegar a uno de los resultados identificados en el punto 1. En el ejemplo de la inserción de clientes un posible camino de ejecución sería: "insertar valores válidos en todos los campos del formulario y pulsar el botón de Insertar". El resultado será, como mencionamos en el punto 1, volver al formulario con un mensaje que nos indique que el cliente se insertó correctamente.

A la hora de identificar los caminos de ejecución hemos de comprobar que cada resultado tenga, al menos un único camino de ejecución que conduzca a él. Si existen resultados para los que es imposible encontrar un camino de ejecución que nos conduzcan a ellos es posible que el cliente no haya definido la historia de uso claramente o la historia abarca demasiada funcionalidad.

Es posible tener varios caminos de ejecución diferentes, e incluso infinitos caminos, con el mismo resultado. También, en este punto, podemos encontrar un camino de ejecución con varios resultados distintos, por ejemplo el camino de ejecución anterior podría terminar con un mensaje de error si el servidor de bases de datos no estuviera disponible. En el apartado 5, una vez identificados los conjunto de pruebas para cada camino, evaluaremos si todos son necesarios o podemos eliminar algunos.

Es posible emplear sencillas notaciones gráficas como grafos de ejecución para simplificar la manera de expresar los caminos de ejecución.

Un valor válido es un valor que, aplicado a un camino de ejecución, nos permite obtener el resultado esperado. Un valor del entorno es algo que no depende del sistema, sino que es externo a él. En nuestro ejemplo de la inserción de clientes, un valor del entorno será el servidor de bases de datos. Según su valor, disponible o no disponible, deberemos obtener un resultado observable diferente a la hora de insertar un cliente.

Podemos encontrar caminos de ejecución redundante cuando tengamos caminos de ejecución iguales, o un camino contenido en otro, que, para los mismos valores genere los mismos resultados. También podemos encontrar caminos de ejecución infinitos especialmente en aquellos que no terminan la historia. Por ejemplo, en la inserción de clientes sería posible insertar clientes correctos infinitamente.

Al final de este proceso el cliente contará con una descripción clara de todas las pruebas de aceptación que debe generar para verificar completamente la funcionalidad resumida en una historia de uso.

5. UN BUEN EJEMPLO.

En este apartado vamos a retomar la historia de uso que se propuso en el punto 2 y vamos a derivar pruebas de aceptación aplicando los pasos vistos en el apartado 4. La historia de uso del apartado 2 se reproduce a continuación, en la tabla 5.

Table 5. Autenticación del médico.

<i>Cuando un médico solicita entrar en el sistema, el sistema solicita su identificador y contraseña. Si el médico está registrado en el sistema accede a la pantalla de gestión de pacientes (RF-02). Si el identificador o la contraseña son incorrectos, el sistema vuelve a solicitarlos.</i>

5.1. Identificar todos los posibles resultados observables de la historia.

A partir de la historia e uso se identifican claramente 2 resultados observables: acceso a la pantalla de gestión de pacientes si la identificación fue correcta y acceso a la pantalla de acceso si la identificación fue incorrecta. Estos resultados se resumen en la tabla 6.

Table 6. Resultados observables de la historia de uso.

Resultados observables.

1	Acceso a la pantalla de gestión de pacientes. Access to patient manager screen.
2	Acceso a la pantalla de acceso. Access to login screen.

5.2. Identificar los resultados que terminan la historia y los que permiten continuar dentro la historia.

La clasificación de los resultados del apartado anterior en terminales, si dan por concluido la historia de uso, o no terminales, si aún seguimos dentro de la historia de uso, se muestra en la tabla 7.

Table 7. Resultados clasificados en terminales o no terminales.

	<i>Resultados observables.</i>	<i>Clasificación</i>
1	Acceso a la pantalla de gestión de pacientes. Access to patient manager screen.	Terminal.
2	Acceso a la pantalla de acceso. Access to login screen.	Non-terminal.

A la vista de los resultados de la tabla 7 se llega a la conclusión de que la única manera posible de terminar con la historia de uso es introducir una identificación válida, o bien que el usuario apague el ordenador. No existiendo ninguna otra manera posible de terminar dicha historia de usuario.

Analizada esta circunstancia en una reunión entre el cliente y los desarrolladores se acordó introducir otra manera de terminar con la historia, consistente en impedir el acceso del usuario si había realizado tres accesos incorrectos. La tabla 8 muestra los resultados clasificados en terminales o no terminales con este nuevo resultado.

Table 8. Resultados clasificados en terminales o no terminales.

	<i>Resultados observables.</i>	<i>Clasificación</i>
1	Acceso a la pantalla de gestión de pacientes. Access to patient manager screen.	Terminal.
2	Acceso a la pantalla de acceso. Access to login screen.	Non-terminal.
3	Acceso a la pantalla de denegación de acceso (después de 3 intentos inválidos). Access to login refused screen (after 3 invalid logins).	Terminal.

5.3. Identificar todos los caminos de ejecución posibles.

En la tabla 9 se muestran todos los posibles caminos de ejecución que llevan a los resultados identificados en los puntos anteriores.

Table 9. Todos los posibles caminos de ejecución

	<i>Caminos de ejecución.</i>	<i>Resultados observables.</i>
1	Introducir un nombre y clave correcto. Write a valid name and password.	Acceso a la pantalla de gestión de pacientes. Access to patient manager screen.
2	Introducir un nombre y clave incorrecto. Introducir un nombre y clave correcto. Write an invalid name and password. Write a valid name and password.	Acceso a la pantalla de gestión de pacientes. Access to patient manager screen.
3	Introducir un nombre y clave incorrecto. Introducir un nombre y clave incorrecto. Introducir un nombre y clave correcto. Write an invalid name and password. Write an invalid name and password. Write a valid name and password.	Acceso a la pantalla de gestión de pacientes. Access to patient manager screen.
4	Introducir un nombre y clave incorrecto.	Acceso a la pantalla de acceso.

	Write an invalid name and password.	Access to login screen.
5	Introducir un nombre y clave incorrecto. Introducir un nombre y clave incorrecto. Write an invalid name and password. Write an invalid name and password.	Acceso a la pantalla de acceso. Access to login screen.
6	Introducir un nombre y clave incorrecto. Introducir un nombre y clave incorrecto. Introducir un nombre y clave incorrecto. Write an invalid name and password. Write an invalid name and password. Write an invalid name and password.	Acceso a la pantalla de denegación de acceso (después de 3 intentos inválidos). Access to login refused screen (alter 3 invalid logins).

Se puede observar como existen varios caminos redundantes, estos caminos se eliminarán en el apartado 5.5.

En esta historia de uso no se han encontrado ningún camino de ejecución posible que no tenga asociado ningún resultado, ni ningún resultado que no tenga un camino de ejecución que conduzca a él.

5.4. Asignación del conjunto de valores.

En este ejemplo no vamos a tener en cuenta posibles valores de entorno.

A la vista de los posibles caminos de ejecución es necesario definir dos conjuntos de valores: una identificación correcta y una identificación incorrecta. Una identificación correcta va a estar compuesta por un nombre y clave válida. Sin embargo, una identificación incorrecta puede estar compuesta por varios conjuntos de valores, como se muestra en la tabla 10.

Table 10. Valores posibles para los caminos de ejecución.

	<i>Conjunto de valores.</i>	<i>Valores concretos.</i>
1	Identificación válida. Valid login.	Nombre: nombrevalido Clave: clavevalida Name: validname. Password: validpassword.
2	Identificación inválida. Invalid login.	Nombre: nombreinvalido. Clave: clavevalida. Name: invalidname. Password: validpassword.
3		Nombre: nombrevalido Clave: claveinvalida Name: validname. Password: invalidpassword.
4		Nombre: nombreinvalido Clave: claveinvalida Name: invalidname. Password: invalidpassword.

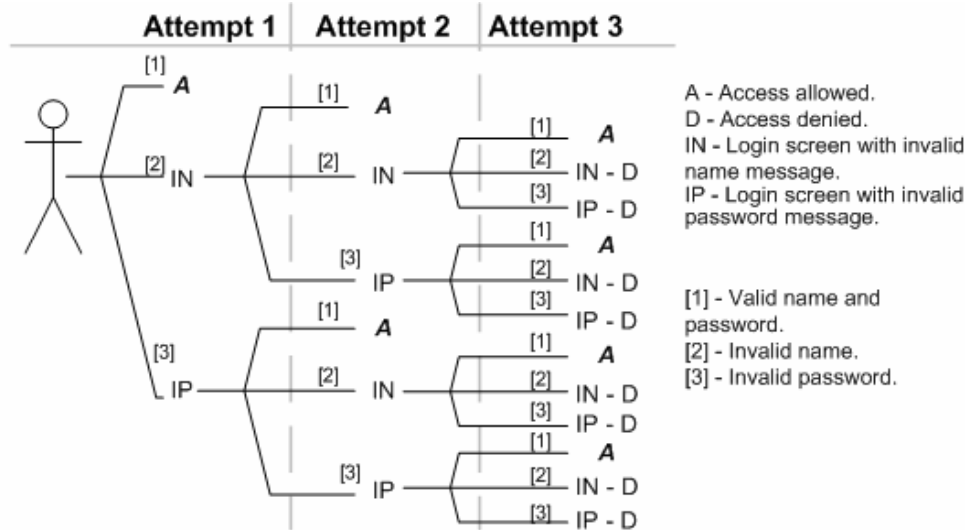
Los valores de la entrada 4 ya están recogidos en las entradas 2 y 3, por lo que no se tendrán en cuenta.

Se observa que para un conjunto de caminos de ejecución se pueden aplicar distintos valores de pruebas para obtener el mismo resultado. Nos planteamos si podemos refinar más aún la funcionalidad de la historia de uso. En la siguiente reunión entre el cliente y los desarrolladores se acordó que el comportamiento del sistema podría ser diferente si se introducía un nombre correcto pero una clave incorrecta. Concretamente se añadió un nuevo resultado consistente en mostrar un mensaje que informara de que nombre era válido pero la clave incorrecta. También se propuso escribir una nueva historia de uso que permitiese que el sistema recordara la clave de un usuario que la hubiese olvidado.

5.5. Eliminación de caminos redundantes

Con los resultados añadidos en el punto anterior y una funcionalidad distinta según se introduzca un nombre inválido y una clave inválida el número de caminos de ejecución aumenta hasta los veintiuno, como se muestra en la ilustración 2.

Ilustración 3. Todos los caminos de ejecución posibles.



Este es un número demasiado elevado para generar una prueba de aceptación a partir de cada camino de ejecución. Para reducirlo vamos a eliminar aquellos casos redundantes.

Por ejemplo, de los veintiún caminos de ejecución posibles, siete de ellos finalizan la historia de uso con el acceso a la gestión de pacientes (introducir el nombre y clave válidos a la primera, segunda o tercera oportunidad). De estos hemos decidido considerar solo los dos primeros (entradas 1 y 2 en la tabla 11).

De los veintiún caminos, ocho conducen a la denegación de acceso, bien por introducir tres nombres inválidos, tres contraseñas inválidas o cualquier combinación posible. De ellos elegimos solo uno con cualquier combinación de nombre / clave posible (entrada 6 en la tabla 11).

De los veintiún caminos seis conducen a un resultado dentro de la historia de uso mostrando de nuevo la pantalla de acceso, bien con un mensaje de nombre inválido o de contraseña inválida. De ellos elegimos solo los dos primeros, uno correspondiente a nombre inválido y otro correspondiente a clave inválida (entradas 4 y 5 en la tabla 9).

La lista definitiva de los seis caminos de ejecución seleccionados junto con sus resultados observables se muestra en la tabla 11.

Table 11. Lista definitiva de caminos de ejecución.

	<i>Camino de ejecución.</i>	<i>Resultados observables.</i>
1	Introducir un nombre y clave correcto. Write a valid name and password.	Acceso a la pantalla de gestión de pacientes. Access to patient manager screen.
2	Introducir un nombre incorrecto. Introducir un nombre y clave correcto. Write an invalid name. Write a valid name and password.	Acceso a la pantalla de gestión de pacientes. Access to patient manager screen.
4	Introducir un nombre incorrecto. Write an invalid name.	Acceso a la pantalla de acceso con mensaje de nombre inválido. Access to login screen with invalid name message.
5	Introducir una clave incorrecta. Introducir un nombre y clave incorrecto. Write an invalid password. Write an invalid name and password.	Acceso a la pantalla de acceso con mensaje de clave incorrecta. Access to login screen with invalid password message.
6	Introducir un nombre incorrecto. Introducir una clave incorrecta.	Acceso a la pantalla de denegación de acceso Access to login refused screen.

Introducir un nombre incorrecto. Write an invalid name. Write an invalid password. Write an invalid name.	
--	--

Estos caminos de ejecución se convertirán en pruebas de aceptación y se automatizarán mediante las herramientas disponibles por el cliente.

6. CONCLUSIONES Y FUTUROS TRABAJOS

Se ha mostrado en este trabajo que una historia de uso es solo un resumen de una funcionalidad que debe tener el sistema. Aunque los desarrolladores van a ir ampliando el conocimiento sobre ese requisito funcional a medida que se implemente, en última instancia es el cliente el que tiene el conocimiento sobre que es lo que quiere y es el que va a comprobarlo mediante una prueba de aceptación para darle el visto bueno. Por ello, es mucho más eficaz que sea el cliente, el cual tiene la información completa quien elabore las pruebas.

También se ha visto que un cliente probablemente no sabrá como escribir pruebas de aceptación que cubran toda la funcionalidad resumida en el caso de uso, por lo que necesitan una guía para desarrollar pruebas de aceptación.

Hemos presentado una propuesta sencilla para guiar a los clientes en el proceso de generación de pruebas de aceptación para una historia de uso. Las ventajas de aplicar esta propuesta son:

- Ayuda a aprovechar el tiempo de los clientes y ayuda a que un cliente se sienta integrado, evitando que se desmoralice por no sabe como preparar pruebas de aceptación.
- El proceso de desarrollo de las pruebas ayuda al cliente a clarificar y concretar la funcionalidad de la historia de uso y favorece la comunicación entre el cliente y el equipo de desarrollo, como se ha visto en el apartado 5.
- El desarrollo de pruebas ayuda identificar y corregir fallos u omisiones en las historias de uso.
- También permite corregir errores en las ideas del cliente, por ejemplo encontrando resultados que el cliente espera encontrar en la implementación pero para los que no existe ningún camino de ejecución que nos conduzca a ello.
- Permite identificar historias adicionales que no fueran obvias para el cliente o en las que cliente no hubiese pensado de no enfrentarse a dicha situación.
- Garantiza la cobertura de la funcionalidad de las pruebas de aceptación, garantizando que no se deja ningún punto importante de la funcionalidad de una historia de uso sin probar.

Una línea de investigación abierta es valorar si merece la pena la automatización de la generación tal y como proponen algunas de las propuestas analizadas en [8]. Hay que tener en cuenta que es difícil automatizar la generación de pruebas de aceptación a partir de las historias de uso ya que estas no recogen una descripción completa y sistemática de la funcionalidad del caso de prueba. Su automatización necesita la definición completa y precisa de toda la funcionalidad resumida en la historia de uso. Para decidir si esta automatización puede ser posible hay que valorar por un lado los beneficios y por otro el gasto de tiempo en definir exhaustivamente la funcionalidad y si el cliente por si solo es capaz de hacerlo o necesita ayuda.

Aunque esta propuesta se centra solo en historias de usuario que recogen funcionalidad, es posible aplicar esta propuesta para desarrollar pruebas de aceptación de otros tipos de requisitos como la fiabilidad del sistema. Un ejemplo de propuesta para desarrollo de pruebas de fiabilidad que podría ser una base para su adaptación a XP se encuentra en [8].

REFERENCIAS

- [1] Beck, Kent. 1999. *Extreme Programming Explained*. Addison-Wesley Professional.

- [2] McBreen, Pete. 2002. *Questioning Extreme Programming*. Addison-Wesley Professional.
- [3] Ken Auer, Roy Miller. 2001. *Extreme Programming Applied*. Addison-Wesley Professional.
- [4] Rick Mugridge, Ewan Tempero. 2003. Retrofitting an Acceptance Test Framework for Clarity. *Agile Development Conference*. EEUU.
- [5] Lisa Crispin, Tip House. 2002. *Testing Extreme Programming*. Addison-Wesley Professional.
- [6] Rick Mugridge, Ewan Tempero. 2003. Retrofitting an Acceptance Test Framework for Clarity.
- [7] Angela Martin, Robert Biddle, James Noble. 2004. The XP Customer Role in Practice: Three Studies. *Agile Development Conference*. EEUU.
- [8] J. J. Gutiérrez, M. J. Escalona, M. Mejías, J. Torres. 2004. Comparative Analysis Of Methodological Proposes To Systematic Generation Of System Test Cases From System Requisites. SV04 Third Workshop on System Testing and Validation.
- [9] Darío Villadiego, María José Escalona, Jesús Torres, Manuel Mejías. 2004. Aplicación de NDT al sistema para el reconocimiento, declaración y calificación del grado de minusvalía. Internal Report.
- [10] Ward Cunningham. Fit: Framework for integrated test. <http://fit.c2.com/>
- [11] Rick Mugridge, Bruce MacDonald, Partha Roop. A Customer Test Generation For Web-Based Systems. *Fourth International Conference on eXtreme Programming and Agile Processes in Software Engineering*. Vol. 1, pp 189-1997.
- [12] Lisa Crispin. 2000. The Need for Speed: Automating Functional Testing in an eXtreme Programming Environment. *QWE2000*. Brussels, BELGIUM.
- [13] Varios autores. 2004. SWEBOK. Guide to the Software Engineering Body of Knowledge. IEEE Computer Society.
- [14] Nebut, C. Fleurey, et-al. 2003. Requirements by contract allow automated system testing. *Proceedings of the 14th International symposium of Software Reliability Engineering (ISSRE'03)*. Denver, Colorado. EEUU.
- [15] Cockburn, Alistair. 2000. *Writing Effective Use Cases*. Addison-Wesley 1st edition. USA.
- [16] Pressman, Roger, 2004. *Ingeniería del Software. Un Enfoque Práctico*. Sexta edición Prentice-Hall.
- [17] Jacobson I., Booch G., Rumbaugh J. 2000. *El Proceso Unificado de Desarrollo de Software*. Addison-Wesley. USA.