MÉTODOS DE TESTING SOBRE LA INGENIERÍA DE REQUISITOS WEB DE NDT

Maria José Escalona
Departamento de Lenguajes y Sistemas Informáticos.
Universidad de Sevilla
mjescalona@us.es

Manuel Mejías
Departamento de Lenguajes y Sistemas Informáticos.
Universidad de Sevilla
risoto@lsi.us.es

Javier Jesús Gutiérrez
Departamento de Lenguajes y Sistemas Informáticos.
Universidad de Sevilla
escalona@lsi.us.es

Jesús Torres
Departamento de Lenguajes y Sistemas Informáticos.
Universidad de Sevilla
jtorres@lsi.us.es

RESUMEN

Las técnicas de testing son un recurso necesario para asegurar la calidad de los resultados de un proyecto software. Esta necesidad de la aplicación de testing hay que trasladarla a la ingeniería web, donde, además, merece la pena proponer técnicas que permitan su aplicación en las primeras fases del ciclo de vida.

Este trabajo presenta una visión global de una propuesta metodológica para la web, NDT (Navigational Development Tecniques) que trabaja en las primeras fases del ciclo de vida y cómo las técnicas de testing pueden aplicarse a esta propuesta en la fase de ingeniería de requisitos.

PALABRAS CLAVES

, Testing, Ingeniería de Requisitos, Validación y verificación de requisitos

1. INTRODUCCIÓN

El campo de la ingeniería web [4] cada día está tomando más interés dentro de la ingeniería del software. El gran número de propuestas metodológicas que existen van encaminadas a ofrecer un marco de referencia adecuado para el equipo de desarrollo de sistemas web. Sin embargo, tras realizar diferentes estudios [1][8] y estudiar trabajos comparativos [5][10][12] se puede llegar a la conclusión de que la etapa de la ingeniería de requisitos para sistemas web es aún un campo poco trabajado y que actualmente comienza a nacer. Con esta motivación nace la propuesta NDT (Navigational Development Techniques)[6][7]. Centrada en las primeras fases del ciclo de vida, NDT propone un entorno metodológico que permite realizar la especificación de requisitos y el análisis de sistemas web de una manera sistemática.

Sin embargo, en el momento en el que se plantea una ingeniería de requisitos en cualquier propuesta es necesario considerar un aspecto importante que es el de testing. La necesidad de garantizar la calidad del producto que se está desarrollando ya desde las primeras fases del ciclo de vida obliga de manera directa al uso de técnicas de testing que garanticen el buen resultado del proyecto.

En este artículo se presenta una visión general de NDT en el apartado 2. En el apartado 3 se hace un breve resumen del resultado de un estudio de los diferentes tipos de pruebas de testing y centrándose sobre la ingeniería web y los requisitos funcionales en concreto. Esta introducción al estado del arte en este entorno permite, ya en el apartado 4, plantear un caso práctico en NDT para ver el resultado de estas técnicas.

2. VISIÓN GENERAL DE NDT

Cuando se comienza el desarrollo de un sistema software, sobre todo si es un sistema software de gran tamaño, complejo y susceptible de sufrir cambios en el futuro, es necesario plantear el desarrollo mediante un marco de referencia metodológico que garantice la calidad de los resultados. Cuando este sistema es un sistema Web, hay que buscar ese marco de referencia en las metodologías para la Web. Existen una gran cantidad de propuestas web: UWE[11], WebML[3], OOHDM [13], OOWS [9] u OO-H [2] son sólo algunos ejemplos. Sin embargo, estas propuestas están principalmente centradas en fases tardías, diseño e implementación, del ciclo de vida. Existe por tanto la necesidad de encontrar marcos de referencias adecuados para trabajar en las primeras fases. NDT (Navigational Development Techniques)[6][7] es un proceso metódico que se centra en las fases de requisitos y análisis y ofrece una guía sistemática para tratar en estas fases con los aspectos de navegación e interfaz abstracta.

Si se realizan y analizan estudios comparativos [1][5][10][11], se puede concluir que ya existen suficientes lenguajes de modelado y modelos en análisis que han resultado válidos para modelar la navegación y la interfaz abstracta. Por ello NDT utiliza lenguajes de modelo estándar ya aceptados por la comunidad investigadora.

La aportación más importante de NDT es que ofrece una guía sistemática para el tratamiento de la navegación y la interfaz. En este sentido, se podría indicar que NDT es una propuesta orientada al proceso. NDT describe de manera detallada todos los pasos que hay que realizar para tratar los requisitos y a partir de ellos conseguir los modelos de análisis. Por otro lado, es una propuesta orientada a la técnica. En todo el proceso propuesto por NDT se indica qué técnicas hay que usar, el modelo de aplicación y el resultado que hay que obtener. Y, por último, es una propuesta orientada al resultado. Tras la aplicación de las técnicas se consiguen resultados y modelos cuya nomenclatura y estructura está completamente detalladas en NDT. Además, tras la aplicación de todo el proceso, en NDT se obtienen una serie de resultados generales: el documento de requisitos del sistema, el documento de análisis del sistema y los prototipos de la interfaz abstracta. La estructura de todos ellos está descrita en NDT[6][7].

2.1.1 Ciclo de vida de NDT

El ciclo de vida de NDT está compuesto por dos fases: la ingeniería de requisitos y el análisis. Aunque, en principio, ambas son secuenciales, el proceso de NDT no lo es, puesto que en muchos momentos se puede realizar la vuelta atrás para corregir errores o incongruencias.

La fase de ingeniería de requisitos de NDT es una ingeniería de requisitos guiada por objetivos. En la primera etapa de la ingeniería de requisitos se definen cuáles son los objetivos del sistema a desarrollar y en base a ellos se capturan y definen los diferentes requisitos del sistema. Los requisitos en NDT son agrupados según su carácter en requisitos de almacenamiento de información, requisitos de actores, requisitos funcionales, requisitos de interacción y requisitos no funcionales. Cada grupo de requisitos es tratado de una manera particular, adecuada a su tipología.

Una vez capturados y definidos los requisitos se pasa a la validación de los mismos. Si durante la validación se detectan errores, se vuelve a la captura y definición hasta llegar al resultado final adecuado. Este resultado final queda plasmado en el documento de requisitos del sistema.

Con el documento de requisitos, se pasa a la fase de análisis. Durante la fase de análisis se generan varios modelos. El primero de ellos es el modelo conceptual. El modelo conceptual en NDT representa la estructura estática del sistema y viene representado por un diagrama de clases. La generación de este modelo consta de dos partes, la primera de ellas es sistemática y permite conseguir un modelo conceptual básico desde los requisitos. El resultado de este proceso sistemático suele coincidir bastante con el modelo conceptual más adecuado para el sistema, pero por si se pudieran realizar mejoras que aumenten la calidad del resultado, NDT propone una segunda etapa en el proceso de creación del modelo conceptual.

En esta segunda etapa, NDT propone una serie de revisiones en las que el analista debe ir aplicando su experiencia para revisar los resultados del modelo básico. La aplicación de estas revisiones tiene dos ventajas. La primera de ellas es que, a pesar de que NDT ofrezca el proceso sistemático, también deja libertad al analista para aplicar su experiencia. Pero por otro lado, también permite detectar incongruencias y errores cometidos durante la fase de ingeniería de requisitos. Por ello, puede ser posible que durante esta actividad del análisis haya que volver a la ingeniería de requisitos a modificar los resultados.

El segundo modelo que se genera durante el análisis es el modelo de navegación. En NDT el modelo de navegación se compone de una serie de diagramas, con una notación estereotipada a partir del diagrama de clases de UML. Los diferentes diagramas se corresponden a los sistemas de navegación para los diferentes roles de usuario que interactúan con el sistema.

Al igual que en el modelo conceptual, el proceso de generación del modelo de navegación consta de dos partes. La primera de ellas es sistemática y permite conseguir un modelo de navegación básico desde los requisitos. La segunda igualmente consiste en revisar el resultado del proceso sistemático para detectar incongruencias cometidas y para que el analista aplique su experiencia. También durante esta segunda etapa se pueden detectar incongruencias en el resultado de ingeniería de requisitos que puede obligar a volver a la fase anterior para realizar revisiones.

Todos estos cambios que se pueden producir durante la generación del modelo de navegación o del modelo conceptual están controlados y detallados en NDT. NDT ofrece una guía completa de posibles modificaciones e indica cómo afectan a fases y resultados anteriores.

Cuando se tienen el modelo conceptual y de navegación, se genera en NDT la interfaz abstracta. Ésta no viene representada por un diagrama, sino por un conjunto de prototipos evaluables por el usuario.

También durante la evaluación de estos prototipos se pueden detectar errores que obliguen a volver a la etapa anterior.

Todo este proceso se representa en la figura 1 mediante un diagrama de actividades.

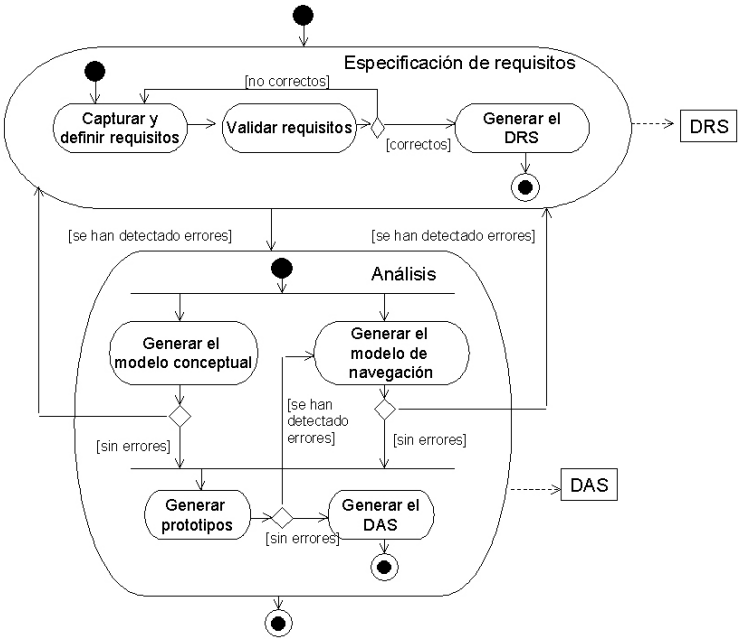

Figura 1. Descripción general de las actividades de NDT.

La aplicación de NDT genera tres resultados finales:

1. El documento de requisitos del sistema, donde se detallan los objetivos y requisitos que debe cumplir el sistema.
2. El documento de análisis del sistema, donde se recogen el modelo conceptual y el modelo de navegación del sistema.
3. Los prototipos del sistema, que muestran la estructura de la interfaz abstracta del sistema.

Estos documentos y prototipos serán la base para la realización de la etapa de diseño e implementación del sistema.

3. MÉTODOS DE TESTING PARA DESARROLLOS WEB.

Hoy en día, debido al aumento del tamaño y la complejidad del software, el proceso de testing se ha convertido en una tarea vital dentro del proceso de desarrollo de cualquier sistema software. Para analizar cómo se pueden aplicar métodos de testing en metodologías web, es necesario realizar una clasificación de los tipos pruebas existentes que permitan analizar la situación actual y cuáles son los tipos de prueba más realizados en desarrollo Web. Durante este apartado se presenta este estudio del arte y la aplicación que de estos métodos se pueden hacer en NDT.

3.1 Tipos de testing.

Existen muchos tipos posibles de pruebas de software. En la tabla1, se recoge una posible clasificación de esas pruebas dependiendo de su cometido y de la fase de desarrollo en que se realizan. Las pruebas unitarias y de integración no pueden realizarse hasta que no se dispone de componentes ya construidos. Las pruebas de implantación y aceptación no pueden realizarse hasta que se tiene el sistema completo y se instala en su entorno de producción. Pero la definición de las pruebas del sistema pueden comenzar antes de que el sistema esté terminado. Como el objetivo de las pruebas de sistema es comprobar que todo lo que se está desarrollando cumple con la funcionalidad recogida en los casos de uso o escenarios, la planificación de estas pruebas y el diseño de los casos de prueba pueden comenzar tan pronto como estén disponibles las especificaciones funcionales.

La planificación y diseño de pruebas de sistema en las primeras fases de desarrollo permiten encontrar errores, omisiones, inconsistencias y sobreespecificaciones en los requisitos funcionales cuando aún es fácil y económico corregirlas.

Tipo de pruebas	**Momento de realización**	**Descripción**
Pruebas Unitarias.	Durante la construcción del sistema	Prueban el diseño y el comportamiento de cada uno de los componentes una vez construidos.
Pruebas de Integración.	Durante la construcción del sistema	Comprueban la correcta unión de los componentes entre sí a través de sus interfaces, y si cumplen con la funcionalidad establecida
Pruebas del Sistema.	Después de la construcción del sistema	Prueban a fondo el sistema, comprobando su funcionalidad e integridad globalmente, en un entorno lo más parecido posible al entorno final de producción.
Pruebas de Implantación.	Durante la implantación en el entrono de producción.	Comprueba el correcto funcionamiento del sistema dentro del entorno real de producción.
Pruebas de Aceptación.	Después de la implantación en el entorno de producción.	Verifican que el sistema cumple con todos los requisitos indicados y permite que los usuarios del sistema den el visto bueno definitivo.
Pruebas de Regresión.	Después de realizar modificaciones al sistema.	El objetivo es comprobar que los cambios sobre un componente, no generan errores adicionales en otros componentes no modificados.

Tabla 1. Clasificación de las pruebas del software

3.2 Testing en el mundo Web.

Las pruebas en desarrollos Web se dividen, tradicionalmente, en pruebas de la parte cliente del sistema y pruebas de la parte servidor del sistema [14]. Las pruebas de cliente tienen como objetivo asegurar que el código HTML es correcto, comprobar el funcionamiento de las partes dinámicas de las páginas, testear que las páginas se visualizan adecuadamente en los navegadores que usan los clientes, pruebas de navegabilidad y usabilidad, etc. Las pruebas de servidor, en cambio, hacen especial hincapié en la seguridad y en la rapidez de las respuestas a los clientes. Sin embargo todas estas pruebas entran dentro de la categoría de pruebas unitarias y pruebas de integración que, como hemos visto, sólo pueden ser hechas una vez que se ha terminado la construcción del sistema. Además, estas pruebas por sí solas no garantizan la calidad del sistema

software, entendiendo aquí la calidad como el cumplimiento por parte del producto de toda la funcionalidad demandada por el usuario. Para comprobar que el sistema efectivamente hace todo lo que recoge el documento de funcionalidad, son necesarias las pruebas del sistema. Esto hace imprescindible la incorporación de procesos sistemáticos de obtención de casos de prueba a la metodología de desarrollo que garanticen que comprobamos toda la funcionalidad acordada para el sistema.

3.3 Métodos de obtención de casos de prueba a partir de requisitos funcionales.

Actualmente existen varias propuestas para automatizar y sistematizar el proceso de obtención de casos de prueba del sistema a partir de los requisitos funcionales que, aunque no han sido desarrolladas específicamente para desarrollos Web, pueden aplicarse a estos desarrollos sin ninguna dificultad, ya que un requisito funcional es independiente del sistema (Web o no) en el que se implemente. En la tabla 2 se incluye un resumen de las cuatro propuestas consideradas.

Aunque cada propuesta hace especial hincapié en algunos aspectos concretos, todas tienen unos principios comunes que enumeramos a continuación:

1. El objetivo de estas propuestas es obtener un conjunto completo de pruebas del sistema que permitan garantizar que el sistema software cumple con la especificación funcional dada, lo cual permite asegurar su calidad.
2. Todas parten de los requisitos funcionales del sistema y todas hacen especial hincapié a comenzar a desarrollar los casos de prueba del sistema en cuanto se dispongan los requisitos funcionales.
3. Todos usan el análisis de los caminos posibles, bien mediante la descripción textual de los pasos del escenario o caso de uso o mediante diagramas de estado.
4. Los requisitos funcionales no tienen que cumplir de principio ningún requisito formal. A partir de una breve descripción en lenguaje corriente ya se puede comenzar a trabajar.
5. La derivación de pruebas del sistema a partir de los requisitos funcionales se realiza de manera automática y sistemática.
6. La aplicación de estas propuestas a los requisitos funcionales ayuda a validarlos, comprobando si son correctos y completos en las primeras fases de desarrollo.

Propuesta.	*Descripción.*
SCENT [17]	*Esta propuesta ofrece un marco de proceso de los requisitos funcionales muy completo, y un proceso basado en los resultados para derivar casos de prueba.*
AGEDIS [19]	*Esta propuesta abarca tanto la generación como la ejecución de las pruebas.*
Generating Test Cases From use Cases.[18]	*Esta propuesta será la estudiaremos con más profundidad en el ejemplo práctico.*
UML-Based Statistical Test Case Generation [19]	*Esta propuesta orientada a obtener casos de prueba para pruebas de carga en función de las probabilidades de uso de los componentes del sistema*

Tabla 2. Propuestas para la obtención de casos de prueba a partir de requisitos funcionales.

4. APLICACIÓN A NDT DE LOS MÉTODOS DE OBTENCIÓN DE CASOS DE PRUEBA.

NDT trabaja con casos de uso y plantillas de descripción de casos de uso en lenguaje natural como herramienta de representación de requisitos funcionales. Esto hace que sea factible aplicar las propuestas del punto anterior a NDT. Vamos a explicar en este punto cómo integrar dichas propuestas dentro de las actividades de NDT y qué beneficios vamos a obtener. No vamos a centrarnos en ningún método concreto sino que trabajaremos sobre la base de las características comunes de todos los métodos estudiados.

La aplicación de los métodos de generación de casos de prueba se incluyen en la actividad de validación de los requisitos funcionales de una manera iterativa. Cuando se obtenga un conjunto de requisitos funcionales completo (todos aquellos que tienen relación entre sí), ya se puede comenzar a derivar los casos de prueba necesarios y a identificar los valores de prueba necesarios.

Como resultado final, obtendremos, además del documento de requisitos del sistema, análisis del sistema y prototipos, un catálogo de pruebas de sistema que permiten comprobar la calidad del sistema, asegurando que el sistema Web cumple la funcionalidad acordada con el cliente.

4.1. Un ejemplo práctico sobre los patrones de NDT

Para desarrollar este ejemplo vamos a seguir la propuesta de Jim Heumann[16] por ser la más sencilla y rápida de aplicar. Los pasos de esta propuesta se resumen en la tabla 3.

Paso	Descripción	Resultado
1	Generar escenarios de uso.	Todos los posibles caminos de ejecución de cada caso de uso. Cada camino es un escenario de uso.
2	Identificar casos de prueba.	Conjunto de casos de prueba a partir de los escenarios anteriores.
3	Identificar los valores a probar.	Valores de prueba asociados a cada caso de prueba anterior.

Tabla 3. Descripción de los pasos para la obtención de un conjunto de pruebas de sistema a partir de casos de uso.

Suponemos que, en la fase de requisitos de NDT, ya se tiene identificado y validado un requisito que es "Entrada en el sistema" cuyo camino de ejecución (la secuencia normal y las excepciones posibles) se muestra en la tabla 4. En NDT cada requisito se define mediante un determinado patrón específico para cada tipo de requisito. En la tabla 4 se muestra un ejemplo del correspondiente a los requisitos funcionales.

RF-1	**Entrar en el sistema**	
Objetivos	OBJ-01: El sistema debe controlar la entrada en el sistema	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando el usuario quiera conectarse al sistema	
Secuencia normal	**Paso**	**Acción**
	1	El usuario conecta con la página de login del sistema.
	2	El sistema solicita autentificación
	3	El sistema recibe la conexión y muestra una página donde pide nombre y contraseña
	4	El usuario rellana el formulario introduciendo su nombre y pulsa el botón entrar
	5	El sistema comprueba el nombre y la contraseña y si ambos son correctos carga la página de inicio
Excepción	**Paso**	**Acción**
	1	[1] Si el servidor no está activo o la página no carga adecuadamente se muestra un mensaje de error y termina el caso de uso.
	4	[2] Si la contraseña o el nombre no se introdujeron el sistema vuelve a solicitarlos indicando un error.
	4	[3] Si el nombre no se encuentra en la lista de nombres registrados el sistema vuelve a solicitarlo indicando un error
	4	[4] Si la contraseña suministrada no es válida, el sistema lo indica con un error y la pide de nuevo.

Tabla 4. Ejemplo de patrón de caso de uso de NDT

Escenarios de caso de uso.	Comienzo	Excepciones	
1. Acceso correcto	Secuencia normal		
2. Fallo de página (servidor no disponible o no carga adecuadamente).	Secuencia normal	Excepción 1	
3. Nombre o contraseña en blanco.	Secuencia normal	Excepción 2	
4. Nombre no existente.	Secuencia normal	Excepción 3	
5. Contraseña no válida(no corresponde con nombre de usuario).	Secuencia normal	Excepción 4	
6. Nombre o contraseña en blanco y después escribir nombre no existente.	Secuencia normal	Excepción 2	Excepción 3
7. Nombre o contraseña en blanco y después escribir clave incorrecta.	Secuencia normal	Excepción 2	Excepción 4
8. Análogo al escenario 6	Secuencia normal	Excepción 3	Excepción 2
9. Nombre no existente y después contraseña incorrecta.	Secuencia normal	Excepción 3	Excepción 4
10. Análogo al escenario 7.	Secuencia normal	Excepción 4	Excepción 2
11. Análogo al escenario 9.	Secuencia normal	Excepción 4	Excepción 3

Tabla 5. Todos los posibles escenarios de uso para el caso de uso en estudio.

La parte más importante de un caso de uso para la generación automática de un caso de prueba es el camino de ejecución. Este camino se divide en dos en el patrón: el camino principal o secuencia normal y los caminos alternativos o excepciones. El camino principal son los pasos que da el sistema si no surge ningún

imprevisto ni error, mientras que los caminos alternativos son las variaciones que pueden surgir en distintos puntos del camino principal a causa de errores, rectificaciones, etc. A cada uno de estos caminos lo llamaremos escenario de caso de uso. Todos los escenarios de caso de uso posibles serán utilizados como base para crear las pruebas.

Un caso de prueba será un conjunto de entradas con datos de prueba, unas condiciones de ejecución, y unos resultados esperados. Para generar los casos de prueba aplicamos los tres puntos de la propuesta:

En el primer punto identificamos todas las combinaciones posibles de caminos de ejecución del caso de uso, es decir todas las combinaciones posibles entre el camino principal y los caminos alternativos y le asignamos un nombre. Cada combinación será un escenario de uso.

Los escenarios de caso de uso del 6 al 11, son redundantes, por los que no los usaremos a la hora de obtener los casos de prueba.

En el segundo punto, estudiamos la descripción del caso de uso de partida y extraemos las condiciones o valores requeridos para la ejecución de los distintos escenarios.

ID	Escenarios	Fallo de página	Nombre	Contraseña	Resultado esperado
1	Acceso correcto	No	V	V	Carga de la página de inicio
2	Fallo de página	Sí	N/A	N/A	Mensaje de error.
3	Nombre o contraseña en blanco.	No	Vacío	Vacío	Mensaje de error. El sistema solicita autentificación de nuevo.
4	Nombre no existente.	No	I	N/A	Mensaje de error. El sistema solicita autentificación de nuevo.
5	Contraseña no válida.	No	V	I	Mensaje de error. El sistema solicita autentificación de nuevo.

I – Valor inválido. V – Valor válido. Vacío – No se indica ningún valor. N/A – El valor que tenga es irrelevante.

Tabla 6. Matriz de casos de prueba.

En el tercer punto, ya con todos los casos de prueba identificados, se revisan para asegurar su exactitud y localizar casos redundantes o la ausencia de algún caso. Por último, para cada escenario de caso de uso, se identifican sus valores de prueba.

ID	Escenarios	Fallo de página	Nombre	Contraseña	Resultado esperado
1	Acceso correcto	HTTP Ok	John	Dough	Carga de la página de inicio
2	Fallo de página	Error 505	N/A	N/A	Mensaje de error.
3	Nombre o contraseña en blanco	HTTP Ok	John “”	“” Dough	Mensaje de error. El sistema solicita autentificación de nuevo.
4	Nombre no existente.	HTTP Ok	Jane	N/A	Mensaje de error. El sistema solicita autentificación de nuevo.
5	Contraseña no válida.	HTTP Ok	John	Anyone	Mensaje de error. El sistema solicita autentificación de nuevo.

Tabla 7. Matriz de casos de prueba con sus valores de prueba.

Con el conjunto escenarios de casos de uso y valores de prueba obtenidos se podrán realizar fácilmente las pruebas del sistema que garanticen que toda la funcionalidad recogida en este caso de uso ha sido correctamente implementada. Repitiendo este proceso por cada caso de uso obtendremos un catálogo con las pruebas del sistema necesarias para garantizar la calidad del sistema desarrollado. Además, generar el catálogo de pruebas en la fase de requisitos de NDT, permite detectar defectos, como caminos alternativos posibles que nos e recogieran en el caso de uso, que, de pasar desapercibidos, obligarían a detener la fase de análisis y retroceder a la fase de requisitos, con el consiguiente aumento de tiempo y coste.

5. CONCLUSIONES Y TRABAJOS FUTUROS

En este trabajo se ha presentado una visión general de NDT y cómo se pueden definir directamente casos de prueba a partir de la definición formal mediante patrones que hace NDT de los requisitos funcionales. Este proceso es susceptible de automatización.

NDT ha sido aplicada en su versión inicial a varios proyectos reales realizados en entornos empresariales [8]. Estos trabajos, ya finalizados, nos están permitiendo evaluar las técnicas en las que estamos trabajando y analizar si, realmente permiten evaluar los resultados. Hasta ahora, los ejemplos realizados están dando

buenos resultados, sin embargo, lo interesante, y en lo que estamos empezando a trabajar es en aplicarlo a ejemplos reales aún no finalizados.

Otro punto abierto importante previsto como trabajo futuro es la incorporación de estas técnicas a NDT-Tool [7]. NDT tiene asociada una herramienta NDT-Tool que permite aplicar sus técnicas y obtener sus modelos y resultados de manera sistemática y/o automática. La idea es que estas técnicas se puedan incorporar a NDT-Tool para que el testing sea lo más automático posible.

Además, hay que tener en cuenta, que las técnicas de testing aplicadas en este trabajo sólo se han incorporado a los requisitos funcionales. La idea futura es incorporar técnicas similares a la aquí presentada para los requisitos funcionales para el resto de requisitos de NDT.

6. REFERENCIAS

1. Barry, C., Lang, M. *A Survey of Multimedia and Web Development Techniques and Methodology Usage*. IEEE Multimedia. 52-56. April-June 2001.
2. Cachero, C. *Una extensión a los métodos OO para el modelado y generación automática de interfaces hipermediales*. PhD Thesis. Universidad de Alicante. Alicante, 2003.
3. Ceri, S. Fraternali, P., Bongio, A., Brambilla M., Comai S., Matera M. *Designing Data-Intensive Web Applications*. Morgan Kaufman. 2003
4. Deshpande, Y., Marugesan, S., Ginige,A., Hanse,S., Schawabe,D., Gaedke, M., White, B. *Web Engineering*. Journal of Web Engineering. Vol. 1 Nº 1. 2002. Rinton Press
5. Escalona, M.J., Torres, J., Mejías, M. *Metodologías de desarrollo de sistemas de información en la web y análisis comparativo*. Novática. Revista De la Asociación de Técnicos de Informática, número 159. Pág. 49-59. Septiembre/Octubre 2002.
6. Escalona M.J, Mejías M, Torres J, Reina A.M. *The NDT Development Process*. Proceedings of IV International Conferences on Web Engineering. ICWE 2003. LNCS 2722. pp. 463-467. Springer Verlag 2003
7. Escalona M.J, Mejías M, Torres J, Reina A.M. *NDT-Tool: A tool case to deal with requirements in web information systems*. Proceedings of IV International Conferences on Web Engineering. ICWE 2003. LNCS 2722. pp. 212-213. Springer Verlag 2003
8. Escalona, M.J., Koch, N. *Requirements Engineering for Web Applications: A Comparative Study*. Journal on Web Engineering, Vol.2 Nº3, pp. 193-212. 2004. Rinton Press
9. Fons, J., Pelechano, V., Albert, M., Pastor, O. *Development of Web Applications from Web Enhanced Conceptual Schemas*. Conference on Conceptual Modeling (ER), Is International, 22nd, 2003 - October, Chicago, Illinois (EE.UU.), Il-Yeol Song, Stephen W. Liddle, Tok Wan Ling, Peter Scheuermann, Springer-Verlag, LNCS, 2813, pp. 232 – 245. Springer Verlag 2003
10. Koch, N. *A Comparative Study of Methods for Hypermedia Development*. Technical Report 9905. Ludwig-Maximilian-University, Munich, Germany.
11. Koch, N. *Software Engineering for Adaptative Hypermedia Applications*. Ph. Thesis, FAST Reihe Softwaretechnik Vol(12), Uni-Druck Publishing Company, Munich. Germany. 2001.
12. Retschitzegger, W. & Schwinger, W. *Towards Modeling of Data Web Applications - A Requirements Perspective*. Proceedings of the American Conference on Informating Systems AMCIS 2000, Vol 1, pp. 149-155. 2000
13. Rossi, G. *An Object Oriented Method for Designing Hipermedia Applications*. PHD Thesis, Departamento de Informática, PUC-Rio, Brazil, 1996.
14. Ash , L. 2003. *The Web Testing Companion: The Insider's Guide to Efficient and Effective Tests*. John Wiley & Sons, Hoboken, USA
15. Offutt , J., Wu, Y. Xiaochen Du and Hong Huang, 2002. Modeling and testing Web-based Applications. *GMU ISE Technical ISE-TR-02-08*. Fairfax, USA
16. Heumann , Jim, 2002. Generating Test Cases from Use Cases. *Journal of Software Testing Professionals*.
17. Ryser, J., Glinz, M. 2003. Scent: A Method Employing Scenarios to Systematically Derive Test Cases for System Test. *Technical Report 2000/03,* Institut für Informatik, Universität Zürich.
18. Riebisch, Philippow, I., Ilmenau, M.G. 2002. UML-Based Statistical Test Case Generation. *Technical University*, Ilmenau, Germany
19. Hartman, A. 2004 AGEDIS Final Project Report *AGEDIS Consortium Internal Report*. http://www.agedis.de/