novática
Revista de la Asociación de Técnicos de Informática

Novática, revista fundada en 1975 y decana de la prensa informática española, es el órgano oficial de expresión y formación continua de **ATI** (Asociación de Técnicos de Informática), organización que edita también la revista **REICIS** (Revista Española de Innovación, Calidad e Ingeniería del Software).

<http://www.ati.es/novatica/>
<http://www.ati.es/reicis/>

ATI es miembro fundador de **CEPIS** (*Council of European Professional Informatics Societies*) y es representante de España en **IFIP** (*International Federation for Information Processing*); tiene un acuerdo de colaboración con **ACM** (*Association for Computing Machinery*), así como acuerdos de vinculación o colaboración con **AdaSpain**, **AI2**, **ASTIC**, **RITSI** e **Hispalinux**, junto a la que participa en **ProInnova**.

Coordinación Editorial, Redacción Central y Redacción ATI Madrid
Plaza de España 6, 2ª planta, 28008 Madrid
Tlfn.914029391; fax.913093685 <novatica@ati.es>
Composición, Edición y Redacción ATI Valencia
Av. del Reino de Valencia 23, 46005 Valencia
Tlfn. 963740173 <novatica_prod@ati.es>
Administración y Redacción ATI Cataluña
Calle Ávila 50, 3a planta, local 9, 08005 Barcelona
Tlfn.934125235; fax 934127713 <secregen@ati.es>
Redacción ATI Andalucía <secreand@ati.es>
Redacción ATI Galicia <secregal@ati.es>
Suscripción y Ventas <novatica.subscripciones@atinet.es>
Publicidad Plaza de España 6, 2ª planta, 28008 Madrid
Tlnf.914029391; fax.913093685 <novatica@ati.es>
Imprenta: Derra S.A., Juan de Austria 66, 08005 Barcelona.
Depósito legal: B 15.154-1975 – ISSN: 0211-2124; CODEN NOVAEC
Portada: "Sueños probatorios" - Concha Arias Pérez / © ATI
Diseño: Fernando Agresta / © ATI 2003

sumario

Nº 224, julio-agosto 2013, año XXXIX

Tema del próximo número: "Empresa 2.0"

Daniel Rodríguez García[1], José Javier Dolado Cosín[2]
[1]Universidad de Alcalá, [2]UAH; Universidad del País Vasco, UPV/EHU; [1, 2]Coordinadores de la sección técnica "Ingeniería del Software" de Novática

<danrodgar@gmail.com>, <javier.dolado@ehu.es>

Presentación
Mejorando el proceso de pruebas de software: Estado del arte

Editores invitados

Daniel Rodríguez García es profesor titular del Departamento de Ciencias de la Computación de la Universidad de Alcalá, licenciado en Informática por la Universidad del País Vasco/Euskal Herriko Unibertsitatea, y doctorado por la Universidad de Reading. Sus intereses se centran en la ingeniería del software, y la aplicación de técnicas de minería de datos y optimización a la ingeniería del software.

José Javier Dolado Cosín es Catedrático de Universidad en el Departamento de Lenguajes y Sistemas Informáticos de la Universidad del País Vasco/Euskal Herriko Unibertsitatea. Sus intereses técnicos giran alrededor de la ingeniería del software, los sistemas complejos y la gestión de proyectos.

La monografía que presentamos a continuación está dedicada a las "pruebas de software". Las pruebas de software constituyen una de las etapas de desarrollo de software que menos interés recibe cuando se define un proyecto de desarrollo de software a pesar de que todo gestor de proyectos es consciente de la importancia de las mismas.

Siendo las actividades de desarrollo más atractivas desde el punto de vista de la creación de aplicaciones, la gestión de las pruebas conlleva un conjunto de actividades no menos interesantes e igual de complejas. Por ejemplo, Panagiotis Louridas [1] menciona algunas de estas tareas, tales como:

- Organizar el conjunto de casos de prueba.
- Asignar los casos de prueba a los responsables de las pruebas.
- Dirigir el proceso de pruebas.
- Recoger los resultados.
- Medir el progreso de las pruebas.

Para llevar a cabo estas tareas disponemos de diversas herramientas y procesos. El número de herramientas de ayuda a la realización de las pruebas es cada vez mayor, disponiendo los desarrolladores de múltiples útiles para probar el código a medida que se va creando. Del mismo modo, también se incrementan las propuestas de procesos o modos de gestionar y realizar las pruebas. Términos como "*test-driven development*" o "*agile testing*" [2] ya son habituales dentro del desarrollo de software y los equipos de desarrollo se adecuan a los nuevos modos de trabajo en los que se integran las pruebas.

También aparecen nuevos enfoques en el desarrollo de software que modificarán nuevamente los modos de realizar las pruebas. Así, recientemente se han descrito los efectos que pueden tener en las pruebas la aplicación de "las reglas de negocio" [3].

Un desarrollo basado en reglas de negocio conlleva, de acuerdo con sus proponentes, una reducción en los problemas de comprensión entre los grupos de "*testers*" y otros grupos de trabajo. También se indica que, debido a la visibilidad de las reglas de negocio, aumenta la comprensión del comportamiento de los programas y permite escribir mejores escenarios de pruebas. Éste es un ejemplo más de cómo el área de las pruebas de software se ve afectada por los modos de desarrollo.

Según los datos publicados en [1] el esfuerzo dedicado a las pruebas (específicamente "*system test*") varía en función del tamaño del proyecto medido en KLOC (miles de líneas de código), dedicando desde un 16% de esfuerzo en proyectos con un tamaño de 1 KLOC hasta llegar a un 29% para proyectos de 500 KLOC.

También el dominio de aplicación es importante, puesto que el ratio nº de "*developers*"/ nº de "*testers*" puede variar desde 20:1 hasta 1:10. El valor más bajo (20:1) corresponde a dominios de sistemas de gestión comerciales y el valor más alto corresponde a los sistemas de control de vuelo de la NASA. Así, tamaño y dominio de la aplicación son factores a considerar junto con los modelos de procesos de desarrollo y de pruebas.

Pero el término que hoy día se escucha por doquier, "*cloud computing*" y que, quizá, cambie los modos de entender la Informática, también tiene sus consecuencias en el modo de entender las pruebas del software. Así lo expresa J. Whittaker, director de ingeniería de Google, que se pregunta sobre los modos en los que las pruebas unitarias, las pruebas de integración y las pruebas de sistema se adaptan al nuevo paradigma de computación en la nube [4][5].

Para alguna de las actividades, estos autores sugieren realizar unos planes de pruebas con una duración de 10 minutos, pero ir modificando la aplicación objetivo de manera continuada. Según su experiencia [5], la concentración de los equipos mejoró y, aunque no se consiguió llegar al límite de los 10 minutos, los equipos consiguieron definir un plan de pruebas en 30 minutos, incluyendo la documentación esencial. Como ellos mismos indican, su experiencia no tiene por qué ser válida para entornos con software de alta seguridad pero es un ejemplo de cómo se puede desarrollar el núcleo de una planificación de las pruebas en muy poco tiempo.

Por otra parte, la realización de las pruebas de software no tiene por qué ser únicamente la actividad del grupo de *testers* tal como las evidencias lo indican en algunas organizaciones.

Así, Mika V. Mäntylä et al. [6] encontraron evidencias en varias organizaciones de que las pruebas no eran una acción realizada únicamente por los especialistas en las mismas. Los miembros de los equipos de trabajo que tenían contacto con los usuarios y clientes mostraron una alta tasa de validación al igual que la tenían los desarrolladores que encontraban defectos. La conclusión de este estudio es que es importante reconocer la diversidad de individuos que realmente participan en las pruebas, y también es importante comprender la relevancia de la validación desde el punto de vista de usuario final.

Desde un punto de vista más técnico, es de destacar el auge de las técnicas de búsqueda aplicadas a la Ingeniería del Software en general y *testing* en particular aplicando técnicas metaherísticas.

Mark Harman et al. [7] han publicado recientemente el estado de la cuestión en este área y mantienen un repositorio con publicaciones relacionadas[1].

La literatura sobre las pruebas de software es amplísima y es imposible abarcar todos los aspectos en un número limitado de páginas. En la monografía que hoy presentamos, encontramos varios artículos escritos por autores punteros en diversas áreas de pruebas de software.

El trabajo realizado por ***Manuel Núñez***, ***Mercedes G. Merayo*** y ***Robert M. Hierons***,

titulado "*Procesos de pruebas basados en modelos: un compromiso adecuado entre teoría y práctica*", aborda la cuestión de la utilización de los modelos formales dentro del proceso de pruebas.

Los autores ***Javier Tuya***, ***Claudio de la Riva***, ***María Jose Suárez-Cabal*** y ***Raquel Blanco*** presentan en su artículo "*Cobertura de consultas SQL y sus aplicaciones*" las cuestiones más relevantes sobre la cobertura de consultas SQL y su aplicación en las pruebas de bases de datos.

En el artículo "*Algoritmos bio-inspirados para la automatización de pruebas software en la industria*", los autores ***Javier Ferrer***, ***Francisco Chicano*** y ***Enrique Alba*** abordan la aplicación de algoritmos de inspiración biológica a las pruebas del software.

El cuarto trabajo de la sección titulado "*Priorización de casos de prueba: Avances y retos*" y realizado por ***Ana Belén Sánchez Jerez***, ***Sergio Segura Rueda*** y ***Antonio Ruiz-Cortés***, presenta una clasificación para la priorización de los casos de prueba.

El último artículo de esta primera sección de la monografía, "*Utilización de MDE para la Prueba de Sistemas de Información Web*" y escrito por ***Federico Toledo Rodríguez***, ***Macario Polo Usaola*** y ***Beatriz Pérez Lamancha***, describe las cuestiones esenciales en la aplicación de la ingeniería dirigida por los modelos a las pruebas de software.

Por último, tenemos la oportunidad de conocer de primera mano, gracias a la reseña escrita por ***Javier Tuya***, los primeros detalles sobre la *norma ISO/IEC/IEEE 29119 - Software Testing.*

El conjunto de artículos que hemos mencionado hasta ahora, cubre una de la parte de las áreas de pruebas del software que en los próximos años tendrán un importante desarrollo.

Por otro lado, este número especial también cuenta con aportaciones desde un punto de vista industrial. ***Tanja E. Vos***, ***Beatriz Marín*** y ***María José Escalona Cuaresma*** presentan un marco metodológico para evaluar ambas técnicas y herramientas de pruebas con la idea de ayudar a las empresas a decidir qué pruebas deben usarse en cada contexto y cómo comparar las distintas herramientas disponibles.

Finalmente, ***Celestina Bianco*** nos presenta un caso de estudio en una industria farmacéutica de la aplicación de métricas para el apoyo de la toma de decisiones en las pruebas de un dispositivo de análisis de sangre para cumplir así con los objetivos de calidad y estándares necesarios.

Referencias

[1] P. Louridas. Test Management. *IEEE Software, 87*, Sept/Oct 2011, pp. 86-91.

[2] L. Crispin, J. Gregory. *Agile Testing: A Practical Guide for Testers and Agile Teams*. Pearson, 2009. ISBN-10: 0321534468.

[3] T. O. Meservy, C. Zhang, E.T. Lee, J. Dhaliwal. The Business Rules Approach and its Effect on Software Testing. *IEEE Software, July/August, 2012*, pp. 60-66.

[4] F. Shull, A Brave. New World of Testing? An Interview with Google's James Whittaker. *IEEE Software, March/April, 2012*, pp. 4-7.

[5] J.A. Whittaker. The 10-Minute Test Plan. *IEEE Software, November/December 2012*, pp. 70-77.

[6] M.V. Mäntylä, J. Itkonen, J. Iivonen. Who tested my software? Testing as an organizationally cross-cutting activity. *Software Quality Journal, 20*:pp. 145-172 (2012). Descargable en PDF desde: <https://wiki.aalto.fi/pages/viewpage.action?pageId=58940614>. Último acceso: 30 de octubre de 2013.

[7] M. Harman, S.A. Mansouri, Y. Zhang. Search-based software engineering: Trends, techniques and applications. *ACM Computing Surveys, 45 (1)*, Artículo 11 (diciembre 2012).

Nota

[1] <http://crestweb.cs.ucl.ac.uk/resources/sbse_repository/>.

Procesos de pruebas basados en modelos: Un compromiso adecuado entre teoría y práctica

Manuel Núñez[1], Mercedes G. Merayo[1], Robert M. Hierons[2]
[1]Departamento de Sistemas Informáticos y Computación, Universidad Complutense de Madrid; [2]Department of Information Systems and Computing, Brunel University Uxbridge, Middlesex (Reino Unido)

<mn@sip.ucm.es>,<mgmerayo@fdi.ucm.es>, <rob.hierons@brunel.ac.uk>

Resumen: *En este artículo presentaremos nuestras contribuciones más recientes en el campo de los procesos de pruebas basados en modelos. Revisaremos las principales características de nuestro trabajo pero omitiremos todos los detalles técnicos. Concluiremos el artículo con una breve discusión sobre la utilidad de formalizar, en la mayor medida posible, los procesos de pruebas y enumerando distintas formas en las que nuestros modelos y resultados se pueden aplicar en entornos industriales.*

Palabras clave: *Procesos de pruebas basados en modelos, sistemas distribuidos, sistemas temporizados.*

Autores

Manuel Núñez es Catedrático de Universidad en la Facultad de Informática de la Universidad Complutense de Madrid, España. Su investigación se centra en el estudio de métodos formales, con un énfasis especial en aproximaciones formales a los procesos de pruebas. Ha participado en más de 100 Comités de Programa y ha publicado más de 130 trabajos en revistas y conferencias internacionales. <http://antares.sip.ucm.es/manolo/>.

Mercedes G. Merayo es Profesora Titular de Universidad en la Facultad de Informática de la Universidad Complutense de Madrid. Su investigación se centra en procesos de pruebas basados en modelos, con un énfasis especial en el análisis de sistemas temporizados. Ha participado en más de 40 Comités de Programa y ha publicado más de 50 trabajos en revistas y conferencias internacionales. <http://antares.sip.ucm.es/mercedes/>.

Robert M. Hierons es *Professor of Computing* en la *School of Information Systems, Computing and Mathematics* de la Universidad de Brunel (Reino Unido). Su investigación se centra en procesos de pruebas basados en modelos, con un énfasis especial en el estudio de sistemas distribuidos, sistemas en tiempo real y sistemas asíncronos. Ha participado en más de 100 Comités de Programa y ha publicado más de 190 trabajos en revistas y conferencias internacionales. <http://www.brunel.ac.uk/~csstrmh/>.

1. Introducción

La creciente complejidad de los sistemas actuales dificulta notablemente la tarea de asegurar altos niveles de confianza en la corrección de las aplicaciones desarrolladas. Este hecho queda patente, con mayor intensidad, en los sistemas software que se usan en áreas donde la seguridad de las personas es un factor crítico (por ej. sistemas de control de tráfico aéreo, centrales nucleares, sistemas de control médico, etc.).

Para paliar esta situación, algunas grandes compañías están tratando de incrementar el grado de formalización a la hora de desarrollar sus sistemas. Por ejemplo, Schneider-Electric y Airbus utilizan el entorno de desarrollo SCADE, basado en el lenguaje formal LUSTRE, para crear software de control de centrales nucleares y el software de control de los modelos Airbus A340/600 y A380, respectivamente. Merece mención aparte Microsoft que durante los últimos años ha puesto un énfasis especial en la aplicación de métodos formales en la producción de su software.

Entre las distintas técnicas utilizadas para incrementar el grado de confianza en la corrección de los sistemas desarrollados podemos afirmar que los *procesos de pruebas* representan la técnica más extendida en entornos industriales.

Brevemente, se puede decir que un proceso de pruebas consiste en comprobar, mediante interacciones sucesivas, que un sistema cumple un conjunto de propiedades y comportamientos deseados. Este tipo de procesos se ha convertido en una parte fundamental del ciclo de vida. De hecho, se ha constituido en un área de trabajo muy próspera que cuenta con la participación activa de una importante comunidad de investigadores y expertos.

Durante la última década se ha implantado una fuerte consciencia sobre la necesidad de aplicar técnicas formales en los procesos de pruebas para conseguir una mayor automatización y, en último término, obtener resultados más fiables. A pesar de todo ello, los procesos de pruebas se realizan generalmente de un modo *artesanal*, decreciendo así los beneficios que podrían obtenerse de la aplicación formal y sistemática de estas técnicas.

Aunque existen numerosas propuestas rigurosas, el alto nivel de abstracción de estos modelos usualmente limita su implantación en entornos industriales. Para poder contrarrestar esta situación, la comunidad científica está realizando un esfuerzo para desarrollar herramientas que implementen las metodologías formales de los procesos de pruebas. Además, algunas compañías punteras dedicadas a la creación de software han introducido en sus procesos de producción propuestas formales para realizar los procesos de pruebas en los sistemas que desarrollan. Entre ellas merece la pena mencionar de nuevo el caso de Microsoft, que mantiene un amplio grupo trabajando en métodos formales para los procesos de pruebas.

En esta línea es especialmente destacable el trabajo [1], en el que investigadores de Microsoft mostraron que el uso de métodos formales aumenta la fiabilidad de los sistemas desarrollados y permite ahorrar en el presupuesto dedicado a los procesos de pruebas, incluso en el caso de utilizar empleados que no tenían conocimientos previos de métodos formales.

Una manera habitual de aumentar la formalización de los procesos de pruebas consiste en utilizar un modelo (usualmente parcial) del sistema a desarrollar y realizar las pruebas partiendo de la información que nos proporciona dicho modelo (el trabajo recopilatorio [2] permite obtener una amplia visión de este campo).

Intuitivamente, los procesos de pruebas basados en modelos consisten en aplicar pruebas al sistema desarrollado y comprobar si el resultado observado es coherente con lo que el modelo establece que debería ocurrir en la situación planteada. Habitualmente estos procesos de pruebas asumen que el sistema que se está analizando es una *caja negra*, es decir, no es posible acceder a su estructura interna (en el caso del software, no se dispone del código fuente) sino que el proceso de

> “Una manera habitual de aumentar la formalización de los procesos de pruebas consiste en utilizar un modelo (usualmente parcial) del sistema a desarrollar y realizar las pruebas partiendo de la información que nos proporciona dicho modelo”

pruebas debe limitarse a interaccionar con el sistema.

Una representación esquemática de estos procesos se presenta en la **figura 1**.

Aunque los procesos de pruebas basados en modelos representan un campo bien establecido, con numerosas propuestas alternativas y con herramientas que sirven para apoyar el uso de las metodologías desarrolladas, la mayoría del trabajo se ha centrado en determinar que el sistema hace de forma correcta lo que se supone que debe hacer.

Sin embargo, en muchos sistemas es especialmente importante asegurarnos no solo de que el sistema hace lo que debe hacer, sino de que lo hace de una determinada forma. En este sentido, merece especial atención el estudio de comportamientos en los que el tiempo juega un papel fundamental.

De esta forma, debemos no solo determinar que el sistema realiza una cierta acción sino que la realiza, por ejemplo, antes de que transcurra una cierta cantidad de tiempo.

En este artículo revisaremos nuestras contribuciones en el campo de los procesos de pruebas basados en modelos para sistemas con información temporal.

Otra de las líneas en la que estamos trabajando, dentro del amplio campo de los procesos de pruebas basados en modelos, consiste en aplicar los modelos clásicos a sistemas con componentes distribuidos. Estos sistemas presentan unas peculiaridades que, desde el punto de vista teórico, se pueden resumir en el hecho de que si no contamos con un centralizador con el que todas las componentes se comuniquen constantemente, entonces no es posible conocer el orden exacto en el que las acciones se produjeron.

El resto de este artículo está estructurado de la siguiente forma. En la **sección 2** presentamos nuestros trabajos en procesos de pruebas para sistemas donde su comportamiento temporal debe ser tenido en cuenta. En la **sección 3** revisamos nuestras contribuciones en el marco de los procesos de pruebas de sistemas que tienen sus componentes distribuidos entre distintas ubicaciones. En la **sección 4** presentamos nuestro trabajo en procesos de pruebas de forma pasiva.

Intuitivamente, un proceso de pruebas es *pasivo* si en lugar de interaccionar con el sistema, se observan los comportamientos de dicho sistema para determinar si son acordes a las propiedades representadas en sus requisitos. Finalmente, en la **sección 5** presentamos nuestras conclusiones y algunas líneas de trabajo en las que se pueden aplicar nuestras metodologías.

2. Procesos de pruebas en sistemas temporizados

2.1. Modelos teóricos

En la actualidad existen numerosos marcos para realizar procesos de pruebas basados en modelos sobre sistemas que presenten requisitos temporales.

Usualmente, estas metodologías utilizan una noción de tiempo *determinista* en el sentido de que los requisitos expresan condiciones de la forma "*después/antes de t unidades de tiempo*".

Sin embargo, en muchas situaciones no es adecuado utilizar una noción tan simple para representar propiedades temporales. En esta línea, la utilización de modelos estocásticos permite especificar propiedades tales como "*con probabilidad p esta acción debería realizarse antes de t unidades de tiempo*". De este modo, el especificador no debe indicar un tiempo exacto para una acción determinada sino una estimación probabilística.

Cuando no se dispone de esta información probabilística, o bien se considera que su inclusión complicaría innecesariamente el

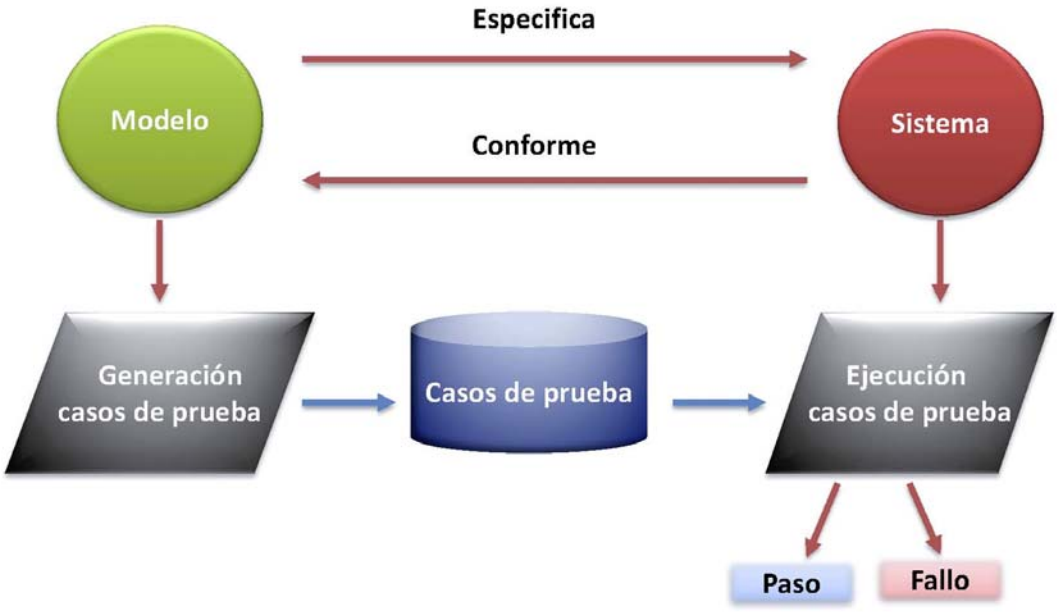

Figura 1. Esquema de proceso de pruebas *activo*.

> “En este punto es donde nuestra metodología para realizar procesos de pruebas basados en modelos con información temporal entra en juego”

modelo, la forma más apropiada para especificar restricciones temporales es el uso de *intervalos de tiempo* que indican el conjunto de posibles valores que se puede asociar con la ejecución de una cierta acción.

Estas disquisiciones han dado lugar al desarrollo de técnicas que se ajustan a las diferentes nociones temporales que se pueden considerar. Desgraciadamente, en la mayoría de los casos dichas técnicas solo son aplicables en un dominio temporal específico, no siendo sencilla su adaptación a sistemas donde se utilizan otros dominios temporales.

En [3] se presenta un marco unificado para especificar y analizar sistemas donde los requisitos temporales se pueden expresar en tres dominios diferentes: *tiempos fijos, variables aleatorias e intervalos de tiempo*.

En primer lugar, es necesario establecer con precisión las condiciones bajo las que se considera que una implementación es correcta con respecto a una especificación. En el caso de modelos deterministas, la noción estándar de corrección es la *equivalencia*: la implementación debe mimetizar el comportamiento de la especificación. Sin embargo, en el caso de que consideremos sistemas no deterministas es más apropiado utilizar una noción de corrección más débil, usualmente denominada *conformidad,* que esencialmente consiste en asegurar que el sistema implementado no *inventa* comportamientos que no estuvieran previstos en la especificación.

En este trabajo se definen varias *relaciones de conformidad*. Todas estas relaciones exigen que el sistema implementado, sin tener en cuenta los aspectos temporales, sea *conforme* con la especificación.

Además, se requiere el cumplimiento de diferentes condiciones temporales, permitiendo que se pueda elegir la más apropiada en cada caso. Por ejemplo, una de las relaciones establece que los tiempos consumidos por la implementación para realizar las acciones sean siempre menores que los indicados en la especificación, mientras en otro caso se exige que sean exactamente los mismos.

Como ya hemos apuntado anteriormente, en el caso de usar un marco temporal *estocástico*, la información se define en términos probabilísticos: la duración de las acciones se expresa en la especificación mediante *variables aleatorias*. Ello permite tener requisitos de la forma "*el tiempo de ejecución de la acción* a *está determinado por la distribución de la variable aleatoria* $\hat{i}_a$". En lo referente a la relación de conformidad para este tipo de sistemas, el hecho de asumir un marco de caja negra impide determinar si las variables aleatorias (desconocidas) asociadas con cada una las acciones del sistema que ha sido implementado están distribuidas de forma idéntica a las correspondientes en la especificación (conocidas). Por ello, es necesario plantear una relación de conformidad más *apropiada*, basada en un conjunto finito de observaciones. La idea consiste en comprobar que los tiempos de ejecución observados son compatibles con las variables aleatorias que aparecen reflejadas en la especificación, estableciéndose dicha compatibilidad por medio de un *contraste de hipótesis*.

2.2. Propuesta metodológica

Hasta ahora hemos discutido la noción de corrección de un sistema con respecto a una especificación que presenta requisitos temporales. Sin embargo, es necesario plasmar en un marco realista los modelos teóricos subyacentes a las relaciones de conformidad. En este punto es donde nuestra metodología para realizar procesos de pruebas basados en modelos con información temporal entra en juego. Idealmente, debemos ser capaces de generar un conjunto de casos de prueba tales que un sistema es conforme con respecto a una especificación si y solo si pasa correctamente todos los casos de prueba.

En [3] presentamos un algoritmo de derivación de casos de prueba que tiene en cuenta los comportamientos que deben analizarse obligatoriamente. Nótese que si no se establece ningún criterio de cobertura, en general tendremos un conjunto infinito de casos de prueba, de forma que suele ser interesante complementar el marco general con técnicas heurísticas que permitan recoger conjuntos representativos de casos de prueba [4].

En un trabajo posterior [5] hemos estudiado en profundidad distintos marcos alternativos para especificar y analizar sistemas con requisitos temporales impuestos mediante intervalos. La principal peculiaridad de este extenso artículo, y que lo distingue del núcleo principal de trabajos en procesos de pruebas para sistemas temporizados, es que permite que un sistema presente *pequeños* errores en su comportamiento temporal. Por ejemplo, si nuestra especificación indica que una acción debe tardar en completarse más de un milisegundo pero menos de tres, y observamos un sistema durante mil interacciones de forma que todas menos dos de ellas caen en el intervalo especificado, entonces podríamos considerar que el sistema es correcto dado que la discrepancia es despreciable y podría deberse, en particular, a la imprecisión de los elementos utilizados para la medida del tiempo.

Por supuesto, serán los requisitos del sistema los que marquen si un error pequeño es admisible o si por el contrario, como en el caso de acciones críticas que tienen que tener un comportamiento exacto en todas las ocasiones, no se permite ninguna desviación del comportamiento esperado.

Los modelos mencionados anteriormente incluyen sólo una noción de paso del tiempo: el tiempo asociado con la ejecución de acciones por parte del sistema. Sin embargo, existen otras situaciones en las que es conveniente tener en cuenta el paso del tiempo cuando se evalúa el comportamiento del sistema. Esta consideración motivó la propuesta de un nuevo marco para llevar a cabo procesos de pruebas [6] en sistemas que pueden evolucionar por medio de *timeouts*.

Esencialmente, un *timeout* es un periodo de tiempo específico durante el cual el sistema permanece a la espera de recibir un estímulo del entorno. Si este tiempo se rebasa y no se ha producido ninguna interacción con el sistema, éste cambia de estado, por lo que sus reacciones a los estímulos recibidos pueden ser diferentes a las que se hubieran producido con anterioridad a dicho límite de tiempo.

Esta consideración complica el marco teórico debido al impacto sobre el comportamiento del sistema, en lo que respecta a las acciones que realiza, de los posibles *timeouts*: una misma secuencia de entradas puede generar diferentes secuencias de salidas dependiendo de los *timeouts* que se hayan producido. Por lo tanto, los aspectos temporales del sistema afectan parcialmente a su comportamiento funcional.

3. Procesos de pruebas en sistemas altamente distribuidos

3.1. Problemática planteada

Los sistemas distribuidos están constituidos por diferentes componentes que se comunican con su entorno a través de diferentes puertos. La principal complicación que pre-

> “En nuestra propuesta hemos considerado que era asumible cuantificar las diferencias potenciales de tiempos entre los diferentes relojes locales y se han investigado diferentes escenarios que han dado lugar a diferentes relaciones de implementación”

sentan estos sistemas durante el proceso de pruebas consiste en integrar apropiadamente la información obtenida por los agentes que realizan las pruebas en cada uno de los puertos. Ello requiere que la interacción de los mismos con el sistema deba realizarse de forma coordinada.

Dado que los puertos pueden llegar a estar en diferentes ubicaciones geográficas, el único modo de sincronizar las interacciones de acuerdo a un plan centralizado requiere una infraestructura de comunicación entre los agentes.

Esta situación puede acarrear un alto coste e incluso interferir con el sistema que se está analizando, pudiendo afectar a los comportamientos observados. Por lo tanto, en lugar de considerar un proceso de pruebas centralizado, es necesario aplicar un plan de interacción para cada puerto en el que, en la medida de lo posible, solo se consideren las acciones que le corresponden y cuyos resultados proporcionen información relevante respecto a la corrección del sistema. Una representación esquemática de esta arquitectura se puede ver en la **figura 2**.

Debido a la encapsulación de estos procesos en cada puerto surgen problemas de *controlabilidad* y *observabilidad*.

El primero de estos problemas corresponde a la dificultad para decidir cuándo se deben aplicar las entradas en cada puerto, debido al desconocimiento de lo que ocurre en los demás puertos. Por ejemplo, supongamos que un caso de prueba comienza con la aplicación de una entrada x en el puerto P, seguida de una salida y en el mismo puerto; a continuación, el agente encargado del puerto Q debe aplicar la entrada z.

El problema que aparece es que desde el puerto Q no se pueden observar las interacciones en el puerto P y, por tanto, no es posible determinar cuándo se debe aplicar la entrada z.

El problema de observabilidad hace referencia a la dificultad para determinar si las observaciones independientes en cada puerto corresponden a un comportamiento correcto del sistema. Esta situación puede llevar a un fallo enmascarado. Supongamos que, como en el ejemplo anterior, un caso de prueba comienza con la aplicación de la entrada x y la emisión de una salida y en el puerto P ; a continuación se debe aplicar nuevamente x y observar la salida y en el mismo puerto P, así como la salida z en el puerto Q. La secuencia inducida por el caso de prueba en el puerto P es $xyxy$ mientras que en el puerto Q deberíamos observar z.

Nótese que las secuencias observadas en ambos puertos serían las mismas si las salidas y y z se produjesen como respuesta a la primera aplicación de la entrada x y tan solo y como respuesta a la segunda entrada. En tal caso, dos fallos simultáneos se enmascararían mutuamente.

3.2. Procesos de pruebas basados en modelos

Los trabajos para definir procesos de pruebas basados en modelos para arquitecturas

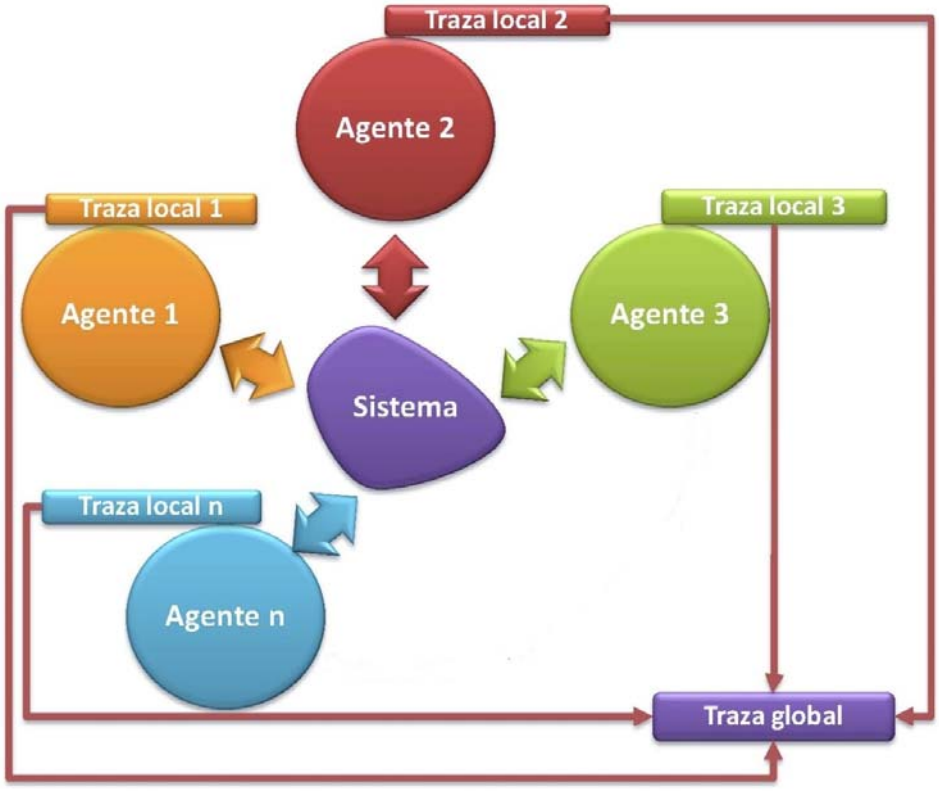

Figura 2. Esquema de la arquitectura para procesos de pruebas en sistemas altamente distribuidos.

> “Los procesos de pruebas *pasivos* parten del hecho de que no se tiene acceso directo al sistema que estamos analizando”

distribuidas se han centrado en sistemas deterministas, usualmente utilizando máquinas de estados finitos deterministas, lo que permite detectar la posible presencia de estos dos problemas en la fase de aplicación de los casos de prueba.

Aunque los formalismos deterministas suelen ser apropiados para modelar sistemas sin componentes, los sistemas distribuidos son frecuentemente no deterministas y por lo tanto estos marcos son poco adecuados. Ello ha llevado al estudio de procesos de pruebas basados en modelos para arquitecturas distribuidas sin asumir el determinismo de los sistemas considerados [7]. En este trabajo se han definido diferentes relaciones de implementación considerando dos posibles escenarios.

El primero de ellos asume que los agentes que llevan a cabo los procesos de pruebas son independientes y tan solo se requiere la consistencia de los comportamientos locales con alguno de los comportamientos establecidos en el sistema global para el puerto correspondiente.

En el segundo caso, existe la posibilidad de que el comportamiento observado en los diferentes puertos pueda analizarse conjuntamente, lo que da lugar a una relación de implementación diferente.

En este trabajo también se proporciona un método para obtener casos de prueba locales, uno para cada puerto, mediante la proyección de un caso de prueba global. Por otra parte, a pesar de que en el caso general el problema de la controlabilidad no es evitable, con el objetivo de reducirlo se restringe el marco a considerar los llamados *casos de prueba controlables*, es decir, casos de prueba que no puede generar una situación en la que el agente local tenga que o aplicar una entrada o esperar una salida, teniendo en cuenta lo que haya ocurrido en otro puerto. Además de caracterizar formalmente los casos de prueba controlables, se presenta un algoritmo para la generación de los mismos a partir de una especificación del sistema que se pretende implementar.

Como continuación de este trabajo se ha desarrollado un nuevo marco [8] para establecer de una forma más precisa la relación causal entre los eventos observados en los diferentes puertos. Para ello se han etiquetado las acciones con información temporal referente al instante en el que se ha producido la observación de las mismas, hecho que permite, en teoría, determinar el orden en el que se han producido las acciones.

Existen dos opciones alternativas. La primera asume la existencia de un reloj global. Al ser esta hipótesis poco realista, la segunda opción considera que en cada puerto se dispone de un reloj local. Si no se dispone de información referente al nivel de sincronización de dichos relojes locales, la información temporal no sería de ninguna utilidad a la hora de ordenar los eventos.

En nuestra propuesta hemos considerado que era asumible cuantificar las diferencias potenciales de tiempos entre los diferentes relojes locales y se han investigado diferentes escenarios que han dado lugar a diferentes relaciones de implementación.

Inicialmente se ha considerado una sincronización perfecta de los mismos, pero dado que esta hipótesis es de nuevo poco realista, y equivale a disponer de un reloj global, se han considerado otras alternativas.

Por una parte se ha asumido que se conoce un límite superior de la posible discrepancia de los relojes, así como que dicha diferencia puede sufrir un incremento lineal. Otra opción que se ha contemplado es que se disponga de una función h que regula dicho crecimiento, de modo que en un tiempo t la diferencia máxima entre los relojes sea $h(t)$. Finalmente, se han estudiado las relaciones entre las diferentes nociones de conformidad propuestas.

4. Procesos de pruebas pasivos

En los marcos de pruebas habituales, como los descritos en las secciones anteriores, se puede interaccionar con el sistema que se está analizando, es decir, es posible aplicar una batería de casos de prueba y observar los resultados. En contraste con este tipo de marcos, podemos mencionar los procesos de pruebas *pasivos* que, aunque con otras denominaciones, son técnicas que aparecen documentadas en la literatura al menos desde finales de los años setenta.

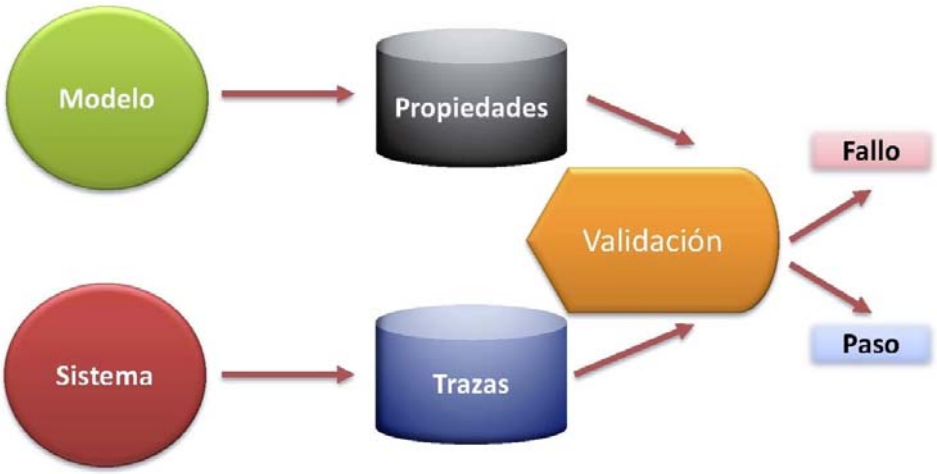

Figura 3. Esquema del proceso de pruebas pasivo.

Estos procesos parten del hecho de que no se tiene acceso directo al sistema que estamos analizando. Esta situación puede darse, por ejemplo, en grandes bases de datos donde se tiene que verificar el buen comportamiento de una funcionalidad y, a diferencia de los procesos de pruebas activos, no se permite la interacción directa con el sistema. Ello implica que no se puedan realizar operaciones como añadir datos, borrar datos, crear tablas, etc.

En este caso, se tiene que estimar la corrección del sistema a partir de la observación de las interacciones realizadas por dicho sistema con un conjunto de datos externos, comprobando si éstas son correctas con respecto a un conjunto de propiedades extraído de la especificación. En otras palabras, un proceso de pruebas pasivo consiste en enfrentar trazas observadas en el sistema con propiedades que dicho sistema debe cumplir en todo momento.

Un ejemplo típico de propiedades que deben verificarse en un sistema es:

Si detectamos un usuario que solicita conectarse al sistema y el sistema le proporcione acceso, entonces si después de realizar una serie de operaciones el usuario solicita la desconexión, ésta debe ser confirmada por el sistema.

En la **figura 3** se muestra un esquema en el que aparecen las principales componentes de un proceso de pruebas pasivo.

En [9] se presenta un marco integrado para realizar un proceso de pruebas pasivo en sistemas con información temporal. En primer lugar, se adaptan las propiedades que se pueden representar de forma que se tengan en cuenta restricciones temporales. Por ejemplo, en el ejemplo anterior podríamos asegurarnos de que el sistema siempre da acceso en menos de cinco segundos.

Un problema importante a la hora de realizar un proceso de pruebas en general es que el número de casos de prueba suele ser astronómico y es necesario establecer un criterio para seleccionar aquellos casos de prueba más relevantes.

En el caso de los procesos pasivos tenemos una situación similar. Por lo tanto, es necesario establecer un mecanismo de selección para determinar que propiedades debemos verificar en primer lugar.

Utilizando un marco basado en *mutación*, hemos analizado distintos tipos de sistemas y de propiedades y nuestras principales conclusiones han sido dos. En primer lugar, las propiedades que revisan pocas acciones del sistema tienden a ser más efectivas a la hora de encontrar acciones erróneas, pero no disponen de la capacidad de encontrar errores por cambios de estado indebidos. En segundo lugar, observamos que el número de fallos detectados crece de forma logarítmica con respecto a la longitud de las trazas observadas.

5. Conclusiones y trabajo futuro

En este artículo hemos revisado algunas de nuestras contribuciones recientes en la extensa área de trabajo conocida como *procesos de pruebas basados en modelos*. El área compagina de manera adecuada metodologías rigurosas, con una fuerte componente formal y matemática, y una visión aplicada, teniendo siempre en cuenta los sistemas reales en los que dichas metodologías se pueden aplicar.

A pesar de que continuamos trabajando en marcos teóricos que extiendan los resultados obtenidos con anterioridad, en la actualidad nuestro principal interés consiste en poner en práctica nuestras metodologías. Nuestros primeros experimentos han sido bastante exitosos y queremos aplicar nuestras propuestas en sistemas más interesantes.

En lo que respecta a los procesos de pruebas para sistemas temporizados, hemos detectado que un enfoque formal permite encontrar errores que no habían sido detectados con anterioridad.

En esta línea, nuestras metodologías son especialmente útiles en pequeños sistemas para los que se ha asegurado su fiabilidad a nivel de las acciones que realizan pero donde los aspectos temporales no se han estudiado con un marco temporizado. Entre este tipo de sistemas podemos destacar electrodomésticos y componentes de sistemas complejos, y donde el factor tiempo es crítico, como es el caso de automóviles.

En lo concerniente a los procesos de pruebas para sistemas altamente distribuidos, además de trabajar en arquitecturas *clásicas* estamos empezando a usar nuestras metodologías en sistemas *cloud*. Para estos sistemas, no solo estudiamos su comportamiento a la hora de analizar las acciones que realizan; nuestros marcos temporizados permiten estudiar propiedades no funcionales relacionadas con aspectos como el rendimiento, análisis coste/beneficio y duración de componentes.

Finalmente, los procesos de pruebas pasivos están resultando útiles para analizar la corrección de sistemas de una forma poco intrusiva con respecto a comportamientos difícilmente reproducibles y con características no funcionales, como es el caso de la seguridad en sistemas críticos.

Reconocimientos

El trabajo presentado en este artículo se ha realizado en el marco de los Proyectos TESIS y ESTuDIo (TIN2009-14312-C02-01 y TIN2012-36812-C02-01).

Referencias

[1] W. Grieskamp, N. Kicillof, K. Stobie, V. A. Braberman. "Model-based quality assurance of protocol documentation: tools and methodology". *Software Testing, Verification & Reliability, vol. 21*, nº 1, pp. 55-71, 2011.

[2] R. M. Hierons et al. "Using formal specifications to support testing". *ACM Computing Surveys, vol. 41*, nº 2, 2009.

[3] M. G. Merayo, M. Núñez, I. Rodríguez. "Formal testing from timed finite state machines". *Computer Networks, vol. 52*, nº 2, pp. 432-460, 2008.

[4] A. Núñez, M. G. Merayo, R. M. Hierons, M. Núñez. "Using genetic algorithms to generate test sequences for complex timed Systems". *Soft Computing, vol. 17*, nº 2, pp. 301-315, 2013.

[5] M. G. Merayo, M. Núñez, I. Rodríguez. "A formal framework to test soft and hard deadlines in timed Systems". *Software Testing, Verification & Reliability, vol. 22*, nº 8, pp. 583-608, 2012.

[6] M. G. Merayo, M. Núñez, I. Rodríguez. "Extending EFSMs to specify and test timed systems with action durations and time-outs". *IEEE Transactions on Computers, vol. 57*, nº 6, pp. 835-844, 2008.

[7] R. M. Hierons, M. G. Merayo, M. Núñez. "Implementation relations and test generation for systems with distributed interfaces". *Distributed Computing, vol. 25*, nº 1, pp. 35-62, 2012.

[8] R. M. Hierons, M. G. Merayo, M. Núñez. "Using Time to Add Order to Distributed Testing". 18th International Symposium on Formal Methods (FM 2012), *Lecture Notes in Computer Science vol. 7436*, 2012.

[9] C. Andrés, M. G. Merayo, M. Núñez. "Formal passive testing of timed systems: theory and Tools". *Software Testing, Verification & Reliability, vol. 22*, nº 6, pp. 365-405, 2012.