Revista
Española de

Innovación, Calidad e Ingeniería del Software

REICIS

Volumen 5, No. 4, diciembre, 2009

Web de la editorial: www.ati.es
Web de la revista: www.ati.es/reicis
E-mail: calidadsoft@ati.es
ISSN: 1885-4486

Copyright © ATI, 2009

Ninguna parte de esta publicación puede ser reproducida, almacenada, o transmitida por ningún medio (incluyendo medios electrónicos, mecánicos, fotocopias, grabaciones o cualquier otra) para su uso o difusión públicos sin permiso previo escrito de la editorial. Uso privado autorizado sin restricciones.

Publicado por la Asociación de Técnicos de Informática (ATI), Via Laietana, 46, 08003 Barcelona.

Secretaría de dirección: ATI Madrid, C/Padilla 66, 3º dcha., 28006 Madrid

Revista Española de Innovación, Calidad e Ingeniería del Software (REICIS)

Editores

Dr. D. Luís Fernández Sanz (director)
Departamento de Sistemas Informáticos, Universidad Europea de Madrid
Dr. D. Juan José Cuadrado-Gallego
Departamento de Ciencias de la Computación, Universidad de Alcalá

Miembros del Consejo Científico

Dr. Dña. Idoia Alarcón
Depto. de Informática
Universidad Autónoma de Madrid

Dra. Tanja Vos
Depto. de Sist. Informáticos y Computación
Universidad Politécnica de Valencia

Dr. D. Alvaro Rocha
Universidade Fernando Pessoa
Porto

Dra. Dña. María Moreno
Depto. de Informática
Universidad de Salamanca

D. Guillermo Montoya
DEISER S.L.
Madrid

Dra. Dña. Antonia Mas
Depto. de Informática
Universitat de les Illes Balears

Dra. Raquel Lacuesta
Depto. de Informática e Ing. de Sistemas
Universidad de Zaragoza

Dr. D. Ricardo Vargas
Universidad del Valle de México
México

Dr. D. José Antonio Calvo-Manzano
Depto. de Leng y Sist. Inf. e Ing.Software
Universidad Politécnica de Madrid

Dña. Mª del Pilar Romay
Fundación Giner de los Ríos
Madrid

Dr. D. Oscar Pastor
Depto. de Sist. Informáticos y Computación
Universidad Politécnica de Valencia

Dra. D. Javier Aroba
Depto de Ing. El. de Sist. Inf. y Automática
Universidad de Huelva

Dr. D. Pablo Javier Tuya
Depto. de Informática
Universidad de Oviedo

Dr. D. José Ramón Hilera
Depto. de Ciencias de la Computación
Universidad de Alcalá

Dra. María José Escalona
Depto. de Lenguajes y Sist. Informáticos
Universidad de Sevilla

Contenidos

REICIS

Editorial **4**

Luís Fernández-Sanz, Juan J. Cuadrado-Gallego

Presentación **5**

Luis Fernández-Sanz

Aplicación de Open HMI Tester como framework open-source para herramientas de pruebas de software **6**

Pedro Luis Mateo, Gregorio Martínez y Diego Sevilla

Comparativa práctica de las pruebas en entornos tradicionales y ágiles **19**

Agustín Yagüe y Juan Garbajosa

Sección Actualidad Invitada:
El futuro estándar ISO/IEC 29119 - Software Testing **33**

Javier Tuya, Universidad de Oviedo

Aplicación de Open HMI Tester como framework open-source para herramientas de pruebas de software

Pedro Luis Mateo Navarro[1,2], Gregorio Martínez Pérez[1], Diego Sevilla Ruiz[2]
[1]Departamento de Ingeniería de la Información y las Comunicaciones
[2]Departamento de Ingeniería y Tecnología de Computadores
Universidad de Murcia, 30.071 Murcia, España
{pedromateo,gregorio,dsevilla}@um.es

Resumen

Las pruebas de software comprenden una fase muy importante del proceso de desarrollo. Este tipo de pruebas tienen como principal objetivo asegurar la calidad, fiabilidad y robustez de un software, dentro de un contexto o escenario donde está previsto que éste sea utilizado. Un subconjunto de estas pruebas corresponde con las que tienen como principal objetivo asegurar el correcto funcionamiento de las interfaces de usuario (o GUIs –*Graphical User Interfaces*–). Este tipo de pruebas de GUI representan un paso crítico antes de que un software sea puesto en funcionamiento y aceptado por el usuario final. Este artículo describe los principales resultados obtenidos como fruto de una serie de investigaciones relacionadas con las pruebas de software y de GUIs, entre los que se encuentra el diseño e implementación de una arquitectura código-abierto utilizada como entorno para el desarrollo de herramientas automáticas de pruebas sobre GUIs.

Palabras clave: Framework para herramientas de test, tests sobre GUIs, generación automática de casos de prueba en GUIs, tests de usabilidad software.

Application of the Open HMI Tester as an Open-source Framework for Software Testing Tools

Abstract

Software testing is a very important phase in the software development process. These tests are performed to ensure the quality, reliability, and robustness of software within the execution context it is expected to be used. Some of these tests are focused on ensuring that the graphical user interfaces (GUIs) are working properly. GUI Testing represents a critical step before the software is deployed and accepted by the end user. This paper describes the main results obtained from our research work in the software testing and GUI testing areas. It also describes the design and implementation of an open source architecture used as a framework for developing automated GUI testing tools.

Key words: Framework for testing tools, GUI testing, GUI test case auto-generation, software usability testing.

Mateo, P.J., Martínez, G., Sevilla, D.,"Aplicación de Open HMI Tester como framework open-source para herramientas de pruebas de software", REICIS, vol. 5, no.4, 2009, pp.6-18. Recibido: 23-7-2009; revisado: 28-10-2009; aceptado: 19-11-2009

 © ATI, 2009

1. Introducción

Las interfaces gráficas de usuario (GUIs) representan un elemento fundamental y crítico de las aplicaciones de hoy en día, llegando a acaparar incluso hasta el 60% del código que se produce en un proyecto software. Por lo tanto, probar la funcionalidad de las GUIs se presenta como una tarea imprescindible para asegurar la calidad, fiabilidad, robustez y usabilidad del sistema completo.

Pese a que está demostrado que la utilización de herramientas avanzadas para las pruebas de GUI permite mejorar los resultados y ahorrar tiempo y recursos a las empresas de desarrollo, la integración de éstas en los desarrollos actuales no es tan frecuente como se podría esperar. El principal motivo es que la naturaleza propia de este tipo de herramientas no facilita su integración en el proceso de desarrollo, ya que suelen poseer características muy específicas del entorno de pruebas para el que inicialmente fueron desarrolladas. Una limitación adicional relacionada directamente con las interfaces gráficas de usuario es que, hoy en día, existe una gran cantidad de sistemas de ventanas que pueden ser empleados en los desarrollos software, por lo que es fundamental el diseño de herramientas adaptables o de propósito general. Todas estas limitaciones, junto con otras tantas, son las que convierten la investigación, el diseño y el desarrollo de herramientas de pruebas abiertas y multiplataforma en un desafío muy interesante.

Este trabajo incluye una breve descripción de la arquitectura Open HMI Tester (OHT) (sección 2) y de algunas aplicaciones reales de ésta en el área de pruebas del software. En la sección 3 se describe una herramienta de captura y reproducción automática de pruebas de software desarrollada sobre la arquitectura mencionada anteriormente. En la sección 4 se explica un sistema de generación automática de casos de prueba para interfaces de usuario. Adicionalmente, en la sección 5 se describen algunas aplicaciones de esta arquitectura en el ámbito de las pruebas de usabilidad. Finalmente, se incluye una sección de conclusiones sobre el trabajo expuesto y algunas referencias sobre el trabajo futuro.

2. Arquitectura Open HMI Tester

La arquitectura *Open-HMI Tester (OHT)* [1] se trata de una arquitectura abierta (no está ligada a ningún sistema operativo ni sistema de ventanas concretos) para pruebas de

interfaz gráfica. Esta arquitectura (figura 1) está compuesta por una serie de módulos que implementan la funcionalidad genérica, y que por lo tanto nunca cambian (representados en la figura por las cajas no coloreadas), y un conjunto de módulos que deben ser adaptados (representados en la figura por las cajas coloreadas) con el fin de dotar a la arquitectura de la funcionalidad necesaria para poder operar sobre un entorno de pruebas (sistema operativo, sistema de ventanas, etc.) concreto.

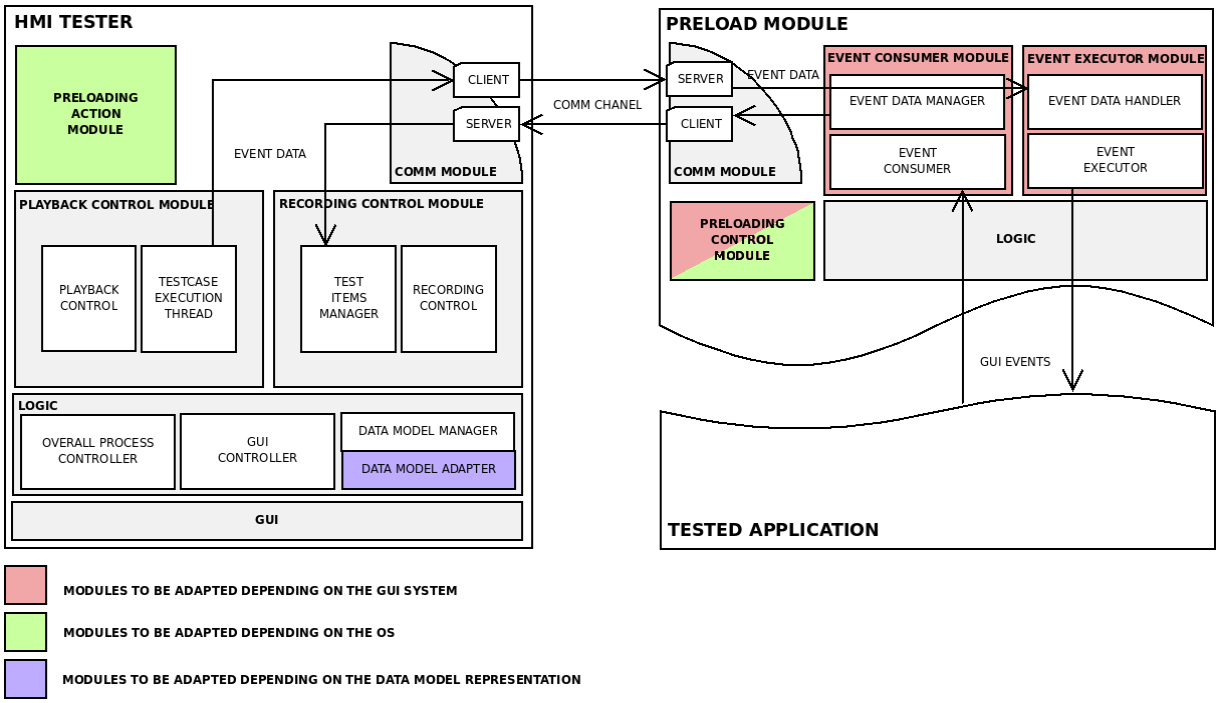

Figura 1: Arquitectura del HMI Tester y del Preload Module.

La arquitectura OHT se divide en dos módulos principales: el primero de ellos es el módulo HMI Tester, el cual se encarga principalmente del control de los procesos; el segundo de ellos corresponde con el módulo Preload, el cual es inyectado en la aplicación testada (en tiempo de ejecución) con el fin de ofrecer servicios de extracción y ejecución de eventos. Ambos módulos tienen la capacidad de comunicarse entre sí, lo que permite a la parte controladora llevar a cabo un proceso de introspección sobre la aplicación testada con dos fines principales: por una parte extraer la información correspondiente a los eventos generados a causa de las acciones que realiza el operador sobre la aplicación a probar; por otra, poder enviar nuevos eventos al núcleo de la aplicación objetivo con el fin de simular, de la manera más realista posible, las acciones del operador sobre la interfaz gráfica de la

aplicación testada. El hecho de que el proceso de introspección se realice mediante la inyección de una librería dinámica en la aplicación testada [2] evita la necesidad de ejecutar ningún tipo de código o de llamada en ésta, lo que permite que las herramientas desarrolladas bajo esta arquitectura puedan ser usadas tanto en aplicaciones en fase de desarrollo como en aplicaciones ya desarrolladas.

El módulo **HMI Tester** tiene como principal cometido el controlar los procesos de grabación (captura de eventos) y de reproducción (ejecución de eventos), así como la gestión completa de la creación y el mantenimiento de los archivos de pruebas (test suites). Como podemos observar en la figura 1, el módulo HMI Tester contiene una serie de submódulos que deben ser adaptados:

- **Data Model Manager and Adapter submodules**: estos submódulos permiten integrar en la arquitectura *Open HMI Tester* la implementación de cualquier representación del modelo de datos. Por lo tanto, deberán ser adaptados dependiendo del modelo de datos seleccionado para representar el conjunto de pruebas (test suite).
- **Preloading Action module**: el principal objetivo de este submódulo es llevar a cabo el proceso de precarga, permitiendo la inyección del módulo *Preload* al lanzar la aplicación a testar. Ya que el proceso de lanzamiento y control de aplicaciones puede incluir acciones concretas del sistema operativo, este módulo deberá ser adaptado dependiendo de las características del entorno de pruebas.

El resto de submódulos incluyen la funcionalidad genérica, y que por lo tanto se reutilizará independientemente de las características del entorno de pruebas en el que estemos trabajando; esta funcionalidad se encarga principalmente de manejar la interfaz de usuario (la correspondiente a la herramienta de pruebas) y las decisiones que se toman en ella, el sistema de comunicaciones hacia el módulo Preload, y el control de los procesos de grabación y reproducción de casos de prueba.

El módulo Preload será inyectado en la aplicación testada como una librería dinámica [2], con el fin de añadir la funcionalidad necesaria para llevar a cabo los procesos de extracción y ejecución de eventos. Su principal cometido será la captura de datos y eventos GUI, y también la ejecución de los eventos y órdenes recibidas desde el módulo

HMI Tester. Como podemos observar en la figura 1, este módulo también contiene una serie de submódulos que deben ser adaptados dependiendo de las características del entorno de pruebas:

- **Preloading Control module**: este submódulo será el encargado de desplegar todos los servicios necesarios para incorporar toda la funcionalidad del módulo Preload a la aplicación testada. Su funcionalidad deberá ser adaptada con el fin de asegurar que el conjunto de servicios necesarios para la introspección (comunicaciones, extracción de datos y ejecución de eventos y acciones) sean desplegados durante el proceso de inicialización de la aplicación testada.
- **Event Consumer module**: este submódulo se encarga de capturar y filtrar los eventos que se generan por las acciones del operador en la aplicación testada, gestionar los datos que éstos contienen y enviarlos hacia el HMI Tester para que sean tratados y almacenados. La adaptación de este submódulo dependerá del sistema de ventanas utilizado y de la jerarquía de eventos que utilice.
- **Event Executor module**: este submódulo ejecutará en la aplicación testada los eventos recibidos desde el módulo HMI Tester. La ejecución de estos eventos permitirá llevar a cabo una reproducción fiel de las acciones del operador que previamente fueron almacenadas, junto con cualquier acción adicional que pueda resultar necesaria para llevar a cabo el control del proceso de ejecución y/o validación. Los nuevos eventos recibidos serán notificados a través de un método de la interfaz. La implementación de este submódulo también dependerá del sistema de ventanas.

El resto de submódulos incluyen la funcionalidad genérica correspondiente a la lógica que gestiona los procesos en la aplicación testada, y la correspondiente al sistema de comunicaciones hacia el módulo HMI Tester.

Como también podemos apreciar en la figura 1, el proceso completo requiere la comunicación entre los dos módulos principales: HMI Tester y Preload. Esta comunicación se llevará a cabo mediante el establecimiento de un canal de datos (por ejemplo, sockets TCP) y el intercambio de elementos de información. Por otra parte, el módulo Preload también deberá establecer una comunicación con la aplicación testada a través de la captura

y envío de eventos, la cual puede ser completada con la ejecución de acciones mediante código.

3. Herramientas de captura y reproducción para GUI

Una de las principales aplicaciones de la arquitectura Open HMI Tester es la creación de herramientas de captura y reproducción para pruebas de interfaces gráficas de usuario. Este tipo de herramientas se centran en capturar la interacción de un operador con la aplicación (normalmente se almacena la secuencia de acciones que el operador ha realizado sobre la GUI), y volcarla a un fichero o soporte similar. Posteriormente, las secuencias de acciones pueden ser recuperadas y reproducidas sobre el software real tantas veces como sea necesario.

Actualmente existen herramientas parecidas que permiten grabar una secuencia de eventos y luego volver a reproducirla. Muchas de estas herramientas tienen la principal desventaja de que no capturan la interacción real del operador, sino que sólo son capaces de acceder a los eventos más externos de la GUI (es decir, clics de ratón y pulsaciones de teclado) y no al resto de los eventos que también forman parte de la ejecución de las acciones. Esta merma en la precisión de las acciones almacenadas puede provocar, por ejemplo, que un mínimo cambio en el entorno de pruebas (por ejemplo, mover la ventana principal de la aplicación 10 píxeles hacia un lado) convierta en inservible todo el esfuerzo realizado en un proceso anterior de grabación de casos de prueba. La baja tolerancia a modificaciones que presentan algunas de estas herramientas provoca que la fiabilidad y robustez de estos sistemas de pruebas quede en entredicho. En el lado opuesto tenemos las herramientas que al igual que ocurre en la arquitectura Open HMI Tester, pueden acceder al núcleo de la aplicación testada gracias a la introspección no intrusiva en código (mediante la inyección de librerías DLL) y por tanto tienen acceso sin problemas a toda la información correspondiente a cualquier evento generado en ella. Esta característica permite llevar a cabo una reproducción de secuencias de eventos en la que realmente se simula la interacción del operador sobre la aplicación, siendo ésta tolerante a ciertas modificaciones no críticas, como por ejemplo, un cambio en la localización de la ventana, la inclusión de nuevos elementos en la GUI, modificaciones en la localización de estos elementos, etc.

Este tipo de herramientas permite llevar a cabo un proceso de pruebas de interfaz gráfica donde el operador es el que decide qué partes de la GUI deben probarse, evitando así la creación de pesados modelos y de enormes baterías de casos de prueba. Esto nos permite solucionar el conocido problema del coverage criteria [3] (o problema del criterio de cobertura en su traducción al español), ya que la responsabilidad de decidir qué partes son testadas y cuáles no recaen completamente sobre el operador. Así, los casos de prueba generados se centrarán en los elementos relevantes y la funcionalidad para la que la GUI fue desarrollada, evitando secuencias de eventos y combinaciones de acciones sin sentido práctico.

Este tipo de herramientas también pueden incluir procesos adicionales (normalmente con el apoyo de herramientas de edición externas o incorporadas en la propia aplicación), como por ejemplo la validación automática mediante la incorporación de eventos especiales, la creación de guías del operador interactivas, o la generación de documentación asociada a las pruebas. Esto permite complementar el proceso de pruebas en base a las necesidades de cada proyecto.

En la figura 2 podemos apreciar una captura de pantalla correspondiente a un prototipo de herramienta de captura y reproducción desarrollado sobre la arquitectura Open HMI Tester. Esta herramienta, que puede ser descargada desde la plataforma Sourceforge en la dirección http://sourceforge.net/projects/openhmitester/, fue implementada con el objetivo de mostrar la funcionalidad y el enfoque aportado por la arquitectura Open HMI Tester. Este prototipo adapta la arquitectura OHT a un entorno de pruebas con las siguientes características:

- Sistema operativo Linux.
- Sistema de ventanas Trolltech Qt4 bajo X-Window.
- Representación del fichero de pruebas con XML.

Este prototipo, desarrollado en lenguaje C++, incluye la funcionalidad básica necesaria para llevar a cabo los procesos de captura y reproducción de eventos sobre aplicaciones desarrolladas bajo las características especificadas más arriba.

Figura 2: Pantalla de una Herramienta de Captura/reproducción desarrollada sobre la arquitectura OHT

4. Generación automática de casos de prueba y procesos de validación sobre GUI

Otra de las áreas en las que se está trabajando es la de la generación automática de pruebas para interfaces de usuario. La solución propuesta en [4] describe un enfoque intermedio entre las pruebas de software basadas en modelos y las que no utilizan ningún tipo de modelado, ya que la solución propuesta se centra principalmente en la creación de un modelo ligero a partir de un conjunto de anotaciones sobre los elementos representativos de la GUI.

El proceso correspondiente a este nuevo enfoque se basa principalmente en dos elementos: el primero de ellos es un conjunto inicial de casos de uso, que será utilizado para describir el comportamiento de la GUI; el otro elemento será un conjunto de anotaciones que describirán las posibles variaciones que pueden afectar a los diferentes elementos que componen la GUI (variaciones sobre los valores contenidos por esos elementos), y adicionalmente una serie de reglas de validación que comprobarán ciertas propiedades de esos elementos.

Como podemos apreciar en la figura 3, una vez hayamos definido un conjunto inicial de casos de uso que describa el funcionamiento de la GUI, el proceso continúa con las siguientes fases:

1. **Anotación de la GUI**: durante este proceso se van anotando los elementos más representativos de la GUI, es decir, los elementos a tener en cuenta en el conjunto de pruebas. Estos elementos pueden ser anotados de dos formas totalmente complementarias: la primera de ellas consiste en indicar un conjunto o rango de posibles valores que un elemento puede tener (cada nuevo valor da lugar a un nuevo caso de prueba); la segunda consiste en comprobar cierto valor de alguna propiedad que deba ser validada (se introducirán nuevas reglas de validación para estos elementos). A la hora de anotar un elemento, el operador podrá escoger cualquiera de las dos alternativas, o las dos en el caso de que sea necesario.

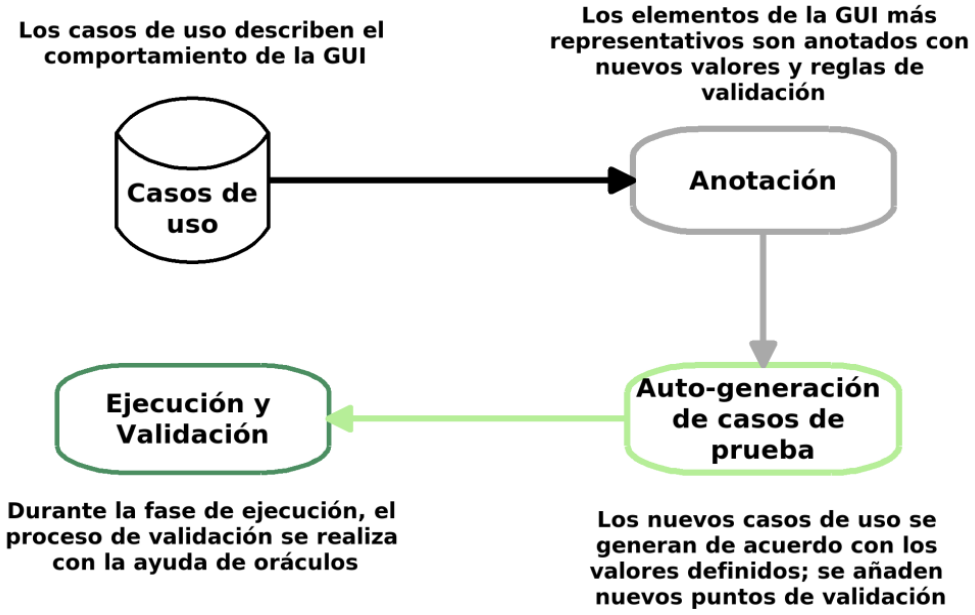

Figura 3: Proceso completo de auto-generación de casos de prueba basado en anotaciones

2. **Generación automática de casos de prueba**: durante esta fase se generará un nuevo caso de prueba para cada posible combinación de valores, es decir, se generarán todos los casos de prueba necesarios para cubrir todo el espectro de valores incluidos en las anotaciones y todas sus combinaciones. También se añadirán nuevos puntos de validación para satisfacer las reglas especificadas en la fase anotación. La figura 4 muestra un ejemplo muy sencillo en el que en la parte izquierda se muestran una serie de anotaciones sobre dos elementos de la GUI, y en la parte derecha se incluye el conjunto equivalente de casos de uso generados. Como podemos apreciar, se han generado cuatro casos de uso para cubrir todas

las posibles combinaciones de los 2x2 valores definidos. Además, se ha añadido un punto de validación para el segundo elemento anotado para poder comprobar las reglas definidas en las anotaciones.

3. **Ejecución y validación**: en este proceso los casos de prueba generados en el paso anterior son ejecutados uno a uno; al mismo tiempo se llevará a cabo el proceso de validación mediante el uso de *test oracles* [5]). Finalmente el proceso devuelve un informe completo en el que se incluirá el resultado, satisfactorio o no, de la ejecución de cada uno de los casos de prueba, y en el que también se incluirá el resultado de cada una de las validaciones.

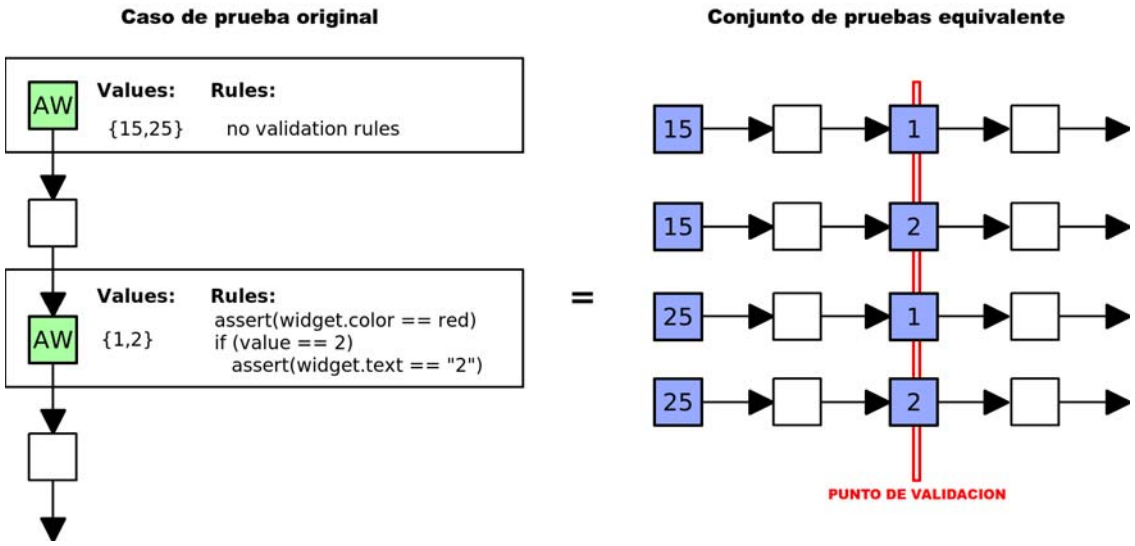

Figura 4: Ejemplo de generación automática de pruebas a partir de anotaciones

Como se ha comentado antes, en la tercera fase se llevará a cabo un proceso de validación basado en *test oracles*, u oráculos de pruebas en su traducción al español. Estos elementos se encargarán de, a partir de la información ofrecida por terceras partes (por ejemplo, una serie de adaptadores para los elementos de la GUI), llevar a cabo el proceso de validación para el que han sido diseñados. Más concretamente, se contempla la posibilidad de incorporar tres tipos diferentes de oráculos: de estado (validan que el estado actual de un elemento sea el mismo que uno previamente almacenado), de validación (validan un conjunto de reglas especificada previamente por el operador), y de *crash* o caída de la aplicación (comprueban si la aplicación ha caído por completo durante la ejecución del caso de prueba).

Esta solución al completo evita el proceso costoso de tener que crear un modelo complejo de la GUI y gran parte de las acciones inherentes a su creación, tales como la verificación, corrección y mantenimiento del modelo, que en la mayoría de los casos implican la intervención manual de las personas que componen el equipo de pruebas. También destacar la solución propuesta al problema del criterio de cobertura, ya que en este caso se evita la inclusión en los casos de prueba de los elementos y propiedades no relevantes mediante la anotación de los elementos más representativos de la GUI. En este caso se puede decir que el criterio de cobertura está dirigido por el usuario u operador. En el lado opuesto también es necesario recordar que ya que el criterio de cobertura queda en manos del operador, y con el objetivo de asegurar un proceso de pruebas completo, éste tiene la obligación de realizar las anotaciones necesarias y suficientes para cubrir todo el funcionamiento y espectro de posibles datos de entrada de la GUI.

Este método nos permite llevar a cabo un proceso de pruebas más ágil, con una mayor escalabilidad y mejor tolerancia a modificaciones respecto a los enfoques centrados en crear un modelo completo; esto se traduce en desarrollos más rápidos, ya que la productividad de los activos aumenta. Este enfoque también permite llevar a cabo un proceso de pruebas iterativo, sobre todo gracias a la posibilidad de reutilizar los casos de uso y las anotaciones. El tamaño del fichero de pruebas puede ir creciendo a la vez que el desarrollo avanza y las pruebas necesitan de un mayor refinamiento.

5. Evaluación de la Usabilidad en GUIs

Recientemente, los esfuerzos sobre la arquitectura *Open HMI Tester* están siendo trasladados hacia otro tipo de pruebas software, más enfocadas hacia la evaluación de la usabilidad de las interfaces gráficas de usuario. De acuerdo a la norma ISO 9241 [6], la usabilidad puede ser vista como “la medida con la que un producto puede ser utilizado por ciertos usuarios con el fin de conseguir sus objetivos con efectividad, eficiencia y satisfacción, en un contexto de uso específico”. Una de las principales motivaciones es que, hasta hace unos años, las investigaciones sobre la evaluación de la usabilidad en el software han estado algo abandonadas, abarcando casi todo el protagonismo la evaluación de la usabilidad en sitios web.

La idea principal es utilizar todo el potencial de la arquitectura *Open HMI Tester* (principalmente la introspección no intrusiva en código) para poder implementar herramientas que permitan automatizar los diferentes procesos de evaluación de la usabilidad. Sin embargo, se cree necesario dotar a la arquitectura de una mayor genericidad con el objetivo de no restringir el funcionamiento general a un proceso de grabación de eventos y a otro de reproducción. Este cambio permitiría al desarrollador añadir su propia lógica de operación, convirtiendo así a la arquitectura *Open HMI Tester* en una arquitectura con un propósito más general, y facilitando así la integración de este tipo de herramientas en el proceso de desarrollo [7].

El hecho de que la lógica de aplicación pueda ser adaptada permite encapsular el conocimiento de los expertos en usabilidad en la propia herramienta de pruebas, facilitando así la automatización completa de los procesos de evaluación. Al mismo tiempo, esta característica facilita la integración de los usuarios finales en el proceso de desarrollo, ya que evita la coincidencia de éstos y los expertos durante la fase de análisis y evaluación.

6. Conclusiones y trabajos futuros

En este trabajo se han presentado algunas de las principales líneas de investigación y desarrollo existentes en la actualidad en relación con los sistemas de pruebas para interfaces gráficas. Se ha presentado la arquitectura *Open HMI Tester*, sobre la que se han diseñado y desarrollado algunas herramientas destinadas a la realización de pruebas automáticas sobre GUIs. Algunos de los ejemplos propuestos son las herramientas de captura y reproducción y la generación automática de casos de prueba basada en anotaciones de los elementos de la GUI. También se ha descrito un nuevo enfoque para generalizar la arquitectura con el objetivo de aumentar el número de submódulos adaptables del *OHT*, donde destaca la posibilidad de adaptar la propia lógica de la herramienta. Con este nuevo enfoque más general se pretende ampliar el abanico de utilidades que pueden desarrollarse sobre esta arquitectura, como por ejemplo herramientas para la evaluación de la usabilidad de un software.

Actualmente se está diseñando una nueva arquitectura más general que permita implementar sobre ella un número mayor de herramientas para las pruebas de software. También se está trabajando para poder hacer realidad algunas de las cuestiones que se

presentan en este trabajo, y que hasta ahora son simples ideas en proceso de maduración, como por ejemplo el desarrollo de una serie de prototipos de herramientas para la evaluación de la usabilidad en las aplicaciones software. Por último, también se está trabajando en otros usos paralelos de la arquitectura, como por ejemplo la posibilidad de extender las pruebas de GUI hacia la lógica de aplicación, o utilizar la arquitectura *OHT* como un elemento independiente para la validación de propiedades y datos de entrada, siguiendo así la filosofía de la programación orientada a aspectos.

Agradecimientos

Este trabajo ha sido parcialmente financiado por la Cátedra SAES de la Universidad de Murcia. Este acuerdo engloba un esfuerzo conjunto entre la empresa SAES (Sociedad Anónima de Electrónica Submarina, http://www.electronica-submarina.com/) y la Universidad de Murcia, para trabajar en software de código abierto y en sistemas de tiempo real y aplicaciones críticas.

Referencias

[1] Mateo Navarro, P., Martínez Pérez, G., Sevilla Ruiz, D., "OpenHMI-Tester: An Open and Cross-Platform Architecture for GUI Testing and Certification", International Journal of Computer Systems Science and Engineering (IJCSSE), Special Issue on Open Source Certification, in press.

[2] Nasika, R., Dasgupta, P., "Transparent Migration of Distributed Communicating Processes", *13th ISCA International Conference on Parallel and Distributed Computing Systems (PDCS)*, Las Vegas (Nevada, USA), November 2000.

[3] Memon, A., Soffa, M., Pollack, M., "Coverage Criteria for GUI Testing", *ESEC/FSE-9: Proceedings of the 8th European software engineering conference held jointly with 9th ACM SIGSOFT international symposium on Foundations of software engineering*, pp. 256-267, New York (USA), 2001.

[4] Mateo Navarro, P., Sevilla Ruiz, D., Martínez Pérez, G., "Automated GUI Testing Validation guided by Annotated Use Cases", *Informatik 2009: Model-based Testing (MoTes09) - 4th Workshop in conjunction with the annual national conference of German Assoc. for Informatics (GI)*, Lübeck (Germany), September 2009.

[5] Xie, Q., Memon, A.M., "Designing and Comparing Automated Test Oracles for GUI-Based Software Applications", *ACM Transactions on Software Engineering and Methodology*, 16, 1, Article 4, 2007.

[6] International Organization for Standardization, "ISO 9241-11 – Guidance on usability", 1998.

[7] Ferré, X., Juristo, N., "How to Integrate Usability into the Software Development Process", *ACM: 28th International Conference on Software Engineering (ICSE 2006)*, Shanghai, China, May 2006.