Revista Española de
Innovación, Calidad e Ingeniería del Software

Volumen 2, No. 2, octubre, 2006

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

Publicado por la Asociación de Técnicos en Informática

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

Contenidos

REICIS

ISSN: 1885-4486

Editorial

REICIS

Con este número REICIS inicia la publicación de trabajos basados en la selección de contribuciones eventos de interés científico y técnico tanto de ámbito nacional e internacional. El proceso incluye la selección de los originales más prometedores para posteriormente solicitar a sus autores una versión mejorada y extendida que es finalmente revisada por el comité editorial. El presente número se ha apoyado en los trabajos presentados en el Taller sobre Pruebas en Ingeniería del Software que fue celebrado en Sitges el 3 de Octubre de 2006 en el marco de las Jornadas de Ingeniería del Software y Bases de Datos.

REICIS va a seguir extendiendo sus fuentes de contribuciones en este tipo de eventos de tanto de ámbito nacional e internacional a través de diversos acuerdos con los comités organizadores de dichos eventos sin descuidar las contribuciones regulares.

En cualquier caso, REICIS continua invitando desde estas líneas a todos los profesionales relacionados con el mundo de la Innovación, Calidad e Ingeniería del Software a que utilicen REICIS como el medio para dar a conocer sus trabajos e investigaciones teniendo las máximas garantías de la profesionalidad con que serán tratados sus trabajos. Podrán encontrar todas las instrucciones necesarias para el envío de sus contribuciones en la página web de la revista: www.ati.es/reicis.

Así mismo REICIS anima a los responsables de eventos técnicos en el ámbito de esta revista a contactar con los editores para posibles acuerdos de publicación de trabajos.

Luis Fernández Sanz
Juan J. Cuadrado-Gallego
Editores

Presentación

REICIS

REICIS reúne en este número dos contribuciones seleccionadas del .

En el primer artículo, el grupo de investigación SYST Research Group de la E.U. Informática de la Universidad Politécnica de Madrid nos presenta la arquitectura TOPEN basadas en es un entorno de validación y operación, diseñado para facilitar el proceso de definición y ejecución de pruebas y procedimientos en sistemas complejos. Este artículo presenta una propuesta para automatizar la generación de un entorno de pruebas a partir de una especificación TTCN-3, tomando como base la arquitectura y funcionalidad de TOPEN.

Javier García, Antonio de Amescua y Manuel Velasco del grupo de Ingeniería de Software del departamento de informática de la Universidad Carlos III de Madrid nos presentan las conclusiones de su análisis sobre los factores que obstaculizan la implantación de los procesos de verificación y validación en organizaciones intensivas en software. Su experiencia en diversos proyectos de mejora de procesos en organizaciones de desarrollo les ha permitido extractar de diez de ellas una serie de importantes problemas que limitan la obtención de todos los beneficios esperables de la mejora de estos procesos de comprobación de software.

Por último se incluye una breve reseña sobre el taller de Pruebas en Ingeniería de Software 2006 (PRIS 2006) que dio origen a los dos trabajos seleccionados para este número de REICIS. La presentación corre a cargo de Javier Tuya, el coordinador de la red de Pruebas en Ingeniería del Software (REPRIS), responsable de la organización del mencionado taller.

Luis Fernández Sanz

Topen como soporte para la automatización de especificaciones en TTCN-3

Pedro P. Alarcón, Agustín Yagüe, Juan Garbajosa y Jessica Díaz
SYST Research Group, E.U. Informática Universidad Politécnica de Madrid (UPM)
Crtra. Valencia Km. 7. E-28031 Madrid Tel: +34 913365083, Fax: +34 913367520
pcavero@eui.upm.es

Abstract

The ever growing complexity of the current software intensive systems requires approaches to increase testing process effectiveness. TTCN-3 is a testing language internationally agreed to define system testing scenarios and their implementation. TOPEN is a validation and operation environment designed to facilitate the definition and execution of tests procedures for complex systems. This paper presents a proposal to automate the generation of a test environment from a TTCN-3 specification, taking TOPEN architecture and functionality as a baseline.

Resumen

La complejidad creciente de los sistemas actuales intensivos en software requiere de técnicas que permitan aumentar la efectividad del proceso de pruebas. TTCN-3 es un lenguaje de pruebas consensuado internacionalmente para definir escenarios de pruebas de sistemas y sus implementaciones. TOPEN es un entorno de validación y operación, diseñado para facilitar el proceso de definición y ejecución de pruebas y procedimientos en sistemas complejos. Este artículo presenta una propuesta para automatizar la generación de un entorno de pruebas a partir de una especificación TTCN-3, tomando como base la arquitectura y funcionalidad de TOPEN.

Palabras clave: Pruebas de aceptación, pruebas de sistema, TTCN-3, Validación, herramientas, entorno de pruebas, automatización de las pruebas

1. Introducción

Los sistemas intensivos en software son cada día más complejos, por lo que la industria requiere de otros enfoques, prácticas y facilidades para probar dichos sistemas. El proceso de validación y las pruebas de aceptación, y de forma relacionada las herramientas, ocupan una

posición cada vez más relevante desde el momento que, en el contexto del conjunto de procesos del ciclo de vida, son muy cercanos a las salidas que el usuario final espera. Si se considera que los procesos del ciclo de vida incluyen una correcta operación de dicho producto, y que las pruebas de validación y/o aceptación de un sistema incluyen procedimientos similares a aquellos para asegurarnos que la operación está siendo correcta, la ejecución sistemática de estas pruebas constituye un aspecto importante para conseguir aumentar la satisfacción del cliente y, probablemente, reducir el coste total del proyecto,

Sin embargo, habitualmente no se presta la atención requerida al proceso de validación. En muchos casos, las pruebas se desarrollan de forma manual, haciendo que el proceso de validación se convierta en tedioso, difícil de gestionar y propenso a errores, lo que puede llevar a no dar suficiente importancia a las pruebas con tal de conseguir cumplir la planificación temporal del proyecto. En otros casos, los equipos de desarrollo producen herramientas de pruebas específicas para sus sistemas o productos, que no son aplicables a otros productos y además suelen tener asociado un alto grado de mantenimiento. Por otro lado las herramientas genéricas de validación, no son siempre fáciles de adaptar a casos concretos. Estas carencias cada vez son más significativas por el tipo de sistemas que se construyen actualmente, sistemas compuestos de diferentes componentes, distribuidos en muchos casos, y con configuraciones y comportamientos que pueden variar dinámicamente.

El Instituto Europeo para Normalización de las Telecomunicaciones (European Telecommunications Standards Institute, ETSI) estandarizó el lenguaje TTCN-3 (Testing and Test Control Notation, tercera versión, referencias [3,4]) con objeto de normalizar la infraestructura de pruebas en el campo de las telecomunicaciones, y está en proceso de aceptación por la industria del sector para validar un gran número de sistemas. TTCN-3 se describe más extensamente en la siguiente sección.

TTCN-3 mantiene una correspondencia casi completa con U2TP (UML 2.0 Testing Profile) definido por OMG [1]. Prácticamente todas las especificaciones de este perfil, con pequeñas excepciones, pueden ser representadas por módulos TTCN-3 y ejecutados en plataformas de pruebas definidas a partir de TTCN-3 [2]. Esta correspondencia permite

avanzar en la generación automática de soluciones TTCN-3 para pruebas a partir de especificaciones UML.

Desde hace varios años el grupo de investigación SYST de la Universidad Politécnica de Madrid viene trabajando en el desarrollo de herramientas software orientadas a la validación y operación de sistemas. Como fruto de estos trabajos, se ha desarrollado un entorno de validación y operación propio, de nombre TOPEN (Test and Operation Environment). TOPEN está basado en una arquitectura de componentes distribuidos y soporta tele-testing. TOPEN puede adaptarse como entorno de pruebas a cualquier sistema que incorpore un interfaz software que permita interacción desde el exterior.

Una cuestión que se plantea siempre que se introduce un nuevo aspecto en un campo tecnológico es su adopción. En el caso del grupo SYST las cuestiones que se plantearon en su momento fueron, primero, cómo podía afectar la presencia de TTCN-3 en TOPEN; segundo, si era conveniente una evolución de TOPEN hacia TTCN-3; y, finalmente, si tenía sentido esta evolución, cómo podía ser posible llevarla a cabo en términos técnicos. Una forma de abordar estas preguntas fue realizar un experimento en el que una especificación TTCN-3, se implementaba considerando que el entorno disponible de pruebas era TOPEN. El enfoque seguido fue utilizar las facilidades que aporta TOPEN para automatizar, en tanto en cuanto fuera posible, la especificación TTCN-3. En este artículo se describen la forma y los diferentes detalles con los que se abordó dicha automatización. En él, primeramente se presentan los aspectos principales de TTCN-3, y posteriormente se explica el entorno TOPEN. Después se describe el enfoque seguido, describiendo la forma en que se interpretó la especificación de TTCN-3 en términos de TOPEN y los aspectos que fue preciso adaptar en éste. Finalmente se incluyen una serie de conclusiones obtenidas en este trabajo.

2. Lenguaje de pruebas TTCN-3

TTCN-3 es un lenguaje de pruebas que permite especificar, ejecutar y automatizar casos de prueba, para pruebas de caja negra de sistemas distribuidos [3]. Es modular y tiene una apariencia similar a la de un lenguaje de programación tradicional. En realidad, TTCN-3 no es un estándar sino un conjunto de estándares [4]: Core Lenguaje, Tabular Presentation Format,

Graphical Presentation Format, Operacional Semantics, TTCN3 Runtime Interface (TRI) y TTCN-3 Control Interface (TCI). Describiremos brevemente el núcleo del lenguaje y el entorno de ejecución de pruebas (TRI/TCI) que componen un sistema de pruebas en TTCN-3.

El bloque principal para construir las especificaciones de pruebas en TTCN-3 es lo que se denomina módulo (una test suite es un módulo). Un módulo TTCN-3 puede definirse con distintas notaciones: formato textual, formato gráfico o formato tabular. Siguiendo [5][6] un módulo es la unidad de compilación y se compone de dos partes:

A. Parte de definiciones:
    A1. Constantes y tipos
    A2. Plantillas (*templates*)
    A3. Definición de componentes y puertos (*port*)
    A4. Declaración de funciones
    A5. Declaración de procedimientos de prueba
B. Parte de control

La declaración de tipos define los tipos de datos utilizados en la comunicación entre los componentes y el sistema a probar, al que se denomina SUT (System Under Test). Estos tipos representan la información intercambiada y procesada por las funciones y procedimientos de prueba. La declaración de plantillas está asociada a un tipo de dato específico y define un conjunto de valores de entrada o de resultados esperados para una prueba determinada.

La declaración de funciones parametrizables especifica el comportamiento de las pruebas. Este comportamiento es implementado usando sentencias y operaciones definidas en el lenguaje TTCN-3. Un tipo especial de funciones son los procedimientos de prueba, que obtienen siempre un valor de tipo "veredicto". Finalmente, la parte de control gestiona la ejecución de los procedimientos de prueba. Éstos se invocan mediante la función "execute" y para cada ejecución se determina un "veredicto", que representa si el procedimiento de prueba ha tenido éxito, fallo o cualquier otro tipo de mensaje. Los procedimientos de prueba se describen en la parte de definición del módulo y se invocan desde la parte de control.

La figura 1 muestra una especificación de una prueba en TTCN-3, planteada en el dominio de un sistema de máquinas de juego interconectadas [19], y que comprueba el

correcto funcionamiento del comando de operación “set” cuando es recibido por una máquina del sistema (objeto myptc en el código del ejemplo). Como puede verse en el ejemplo, si la máquina recibe el comando “set” compuesto por el atributo “creditcounter” y su “valor “ el veredicto de la prueba será éxito, si se recibe cualquier otra cosa o vence el timeout, el resultado de la prueba será fallo.

```
module topenTest {

// A. Module definitions

// A1. Constants and types
type record commandType {
  charstring     attribute,
  charstring     value
}

type record resultType {
  charstring     ok
}

// A2. Templates
template commandType commandTemplate:={
  attribute      := "creditCounter",
  value          := "1000"
}

template resultType resultTemplate:={
  ok     := "ok"
}

// A3. Components and ports
type port commandPortType message {
  out    commandType,
  in     resultType
}

type component ptcType {
  port commandPortType commandPort,
  timer localTimer := 3.0
}

type component systemType {
  port commandPortType commandPort
}
```

```
// A4. Functions
function set() runs on ptcType {
  commandPort.send(commandTemplate);
  localTimer.start;
  alt {
    []commandPort.receive(resultTemplate){
        localTimer.stop;
        setverdict(pass);
      }
    []commandPort.receive {
        localTimer.stop;
        setverdict(fail);
      }
    []localTimer.timeout {
        setverdict(fail);
      }
  }
}

// A5. Test cases
testcase TopenTest_1() runs on ptcType
system systemType {
   var ptcType myptc;
   //create the PTCs
   myptc.create;
   //map the PTCs to the system port
   map (myptc:commandPort,
        system:commandPort);
   //start the PTC's behaviour
   myptc.start(set());
   //wait for the PTCs to terminate
   myptc.done;
 }

// B. Module control

control {
   execute(TopenTest_1());
}

}
```

Figura 1. Especificación de un test en TTCN-3.

Éste es el primer paso para la especificación de un entorno de test abstracto (Abstract Test Suite) siendo necesarios pasos adicionales para alcanzar un entorno de pruebas ejecutable (Executable Test Suite), como la implementación de las interfaces con el SUT (Test Adapter).

Estas interfaces implementarán las operaciones necesarias para adaptar el entorno a la plataforma de ejecución, operaciones como las necesarias para la comunicación con el SUT.

La figura 2 muestra los componentes que forman el entorno de ejecución de TTCN-3. Se identifican tres partes principales en este entorno: el usuario del sistema de test (Test System User), el sistema de pruebas (TTCN-3 Test System) y el SUT [7]. A su vez, el sistema de pruebas se compone de un conjunto de entidades organizadas en tres capas:

- Test Management, que proporciona una interfaz de usuario al administrador.
- TTCN-3 Executable, representa la parte del sistema que interpreta y ejecuta los procedimientos de prueba.
- System Under Test Interface, que proporciona la transformación entre los puertos definidos en el sistema de pruebas y los definidos en el sistema a probar (SUT).

Además, en el sistema de pruebas existen dos interfaces internas TCI (TTCN-3 Control Interface) y TRI (TTCN-3 Runtime Interface) que representan la interacción entre los subsistemas TTCN-3.

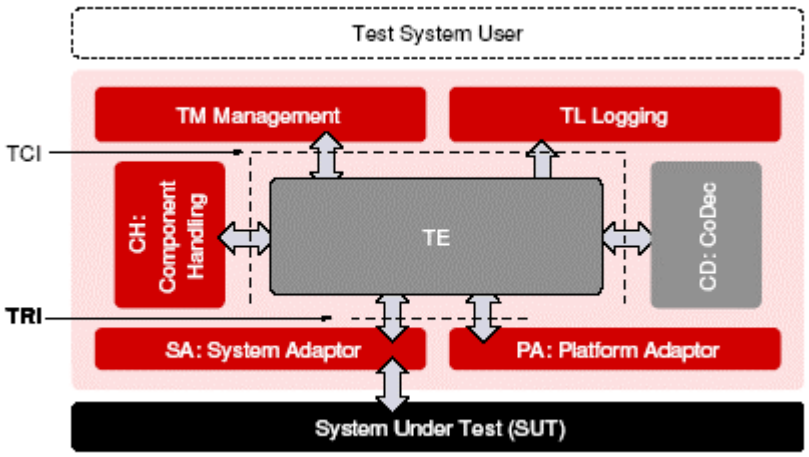

Figura 2. Arquitectura de un sistema de pruebas TTCN-3.

Aunque TTCN-3 tuvo sus orígenes en el campo de las telecomunicaciones, por ejemplo pruebas sobre el protocolo IPv6 [12], está introduciéndose en nuevas áreas entre las que destacan sistemas accesibles a través de servicios web [13] y sistemas embebidos. Las pruebas de componentes de bajo nivel en sistemas de automoción o sistemas aeroespaciales tampoco han quedado fuera del ámbito de TTCN-3.

Existen varias soluciones basadas en el estándar TTCN-3, entre las cuales destacan herramientas comerciales como OpenTTCN Tester[14], Tau Tester[15], TTCN-3 toolbox[16] y TTworkbench[17]. Las tres primeras herramientas soportan la definición de las interfaces TRI y TCI en el lenguaje C, mientras que las dos últimas herramientas indicadas, soportan el lenguaje Java para su definición. Todas ellas soportan el diseño de pruebas a través de su análisis, desarrollo, ejecución y depuración, son independientes del entorno en el cuál se ejecute el SUT y facilitan al usuario la automatización y comprensión de la ejecución de pruebas a través de la visualización del tráfico de mensajes entre el SUT y el operador de pruebas. Algunas de ellas, como TTworkbench presentan funciones de análisis de casos de prueba y generación de documentación en formato HTML ([18]). Por último, mencionar el esfuerzo realizado para que editores de texto convenciones como UltraEdit, Judit o Emacs incluyan la sintaxis TTCN-3. Sin embargo no presentan las ventajas de TOPEN en cuanto a una arquitectura flexible que aúna pruebas y operación remotas y que además permite trabajar con la base de datos de los resultados de las ejecuciones de las pruebas.

3. El entorno de operación y validación TOPEN

TOPEN es un entorno de pruebas y operación de sistemas complejos que permite realizar pruebas de aceptación a sistemas que posean una interfaz software de acceso al sistema. Estas pruebas, lógicamente, son pruebas de caja negra. Una descripción de las funcionalidades de TOPEN se puede encontrar en [8] y [9]. TOPEN Proporciona a los ingenieros de pruebas un marco para definir, compilar y ejecutar procedimientos de pruebas sobre el SUT, y almacenar la información de los procedimientos de prueba y el resultado de sus ejecuciones. La definición de un procedimiento de prueba (TP, *test procedure)*, se realiza mediante un lenguaje de alto nivel, cercano al ingeniero de pruebas, descrito con una sintaxis orientada al dominio de aplicación. Un TP está compuesto de una serie de comandos de operación, que incluye comandos para especificar los valores de entrada, el procesamiento de la prueba y los resultados esperados. Los comandos de un TP deben seguir la gramática previamente definida para

el dominio de aplicación del SUT a validar. Los datos almacenados relativos a los resultados de las ejecuciones de los TPs permiten obtener el veredicto de las pruebas realizadas, e incluso validar algunos requisitos que requieren la evaluación del resultado de varias ejecuciones de un mismo TP, o del resultado de un conjunto de TPs. Esta información también facilita la realización de pruebas de regresión.

La arquitectura TOPEN, representada en la parte central de la figura 3, comprende los siguientes componentes:

- MMI (*Man Machine Interface*), constituye el interfaz gráfico de usuario desde el que el ingeniero de pruebas visualiza el estado del SUT en cada momento, y soporta la definición/ejecución de procedimientos de prueba.
- TE (*Topen Engine*), es el núcleo del entorno TOPEN que se encarga de la compilación y traducción de TPs, y soporta el control de ejecución de las pruebas. El TE soporta la gramática del lenguaje de pruebas del SUT, e incluye la sintaxis y comportamiento de los comandos de operación y sentencias típicas de un lenguaje de programación (utilización de variables, control del flujo, etc.).
- MIB (*Mission Information Base*), contiene la base de datos y *reglas de negocio* asociadas.
- Gateway, que se encarga de la comunicación entre SUT y TOPEN (componente TE).

La arquitectura de TOPEN es independiente del dominio de aplicación, lo que permite ser adaptado con un esfuerzo reducido a diferentes sistemas y sus componentes pueden estar distribuidos en diferentes máquinas ya que su diseño lo soporta. Esto facilita que se puedan realizar pruebas remotas (tele-testing). En [10] se ha definido una línea de productos basada en esta arquitectura. Mientras que la base de datos no requiere cambios porque el modelo de datos es independiente del dominio de aplicación [11].

4. La arquitectura de TOPEN como marco para la generación de sistemas de pruebas especificados en TTCN-3

La especificación de sistemas de pruebas en TTCN-3 implica abordar tanto el entorno de pruebas como las pruebas en sí. Ello conlleva la necesidad de tener un profundo conocimiento tanto del lenguaje TTCN-3 como del dominio de aplicación concreto al que se aplique, incluyendo la interfaz con el SUT. De esta forma, es imposible separar el papel de ingeniero de pruebas del de diseñador del entorno de pruebas. Nosotros

consideramos que estos dos papeles deben estar claramente diferenciados, de manera que un ingeniero de pruebas no tenga por qué conocer ni dedicarse al desarrollo de la infraestructura del sistema de pruebas en sí, y su conocimiento se centre en un lenguaje específico de alto nivel para realizar las pruebas.

A continuación describimos los resultados del experimento relativos a la adaptación mencionada. De la especificación de un módulo TTCN-3, se puede obtener la información necesaria para adaptar TOPEN al SUT correspondiente. Esta adaptación en términos de TOPEN se denomina generación pues éste es el proceso que se sigue mayoritariamente a partir de unos módulos predefinidos.

La parte derecha de la figura 3 describe gráficamente qué partes de un módulo TTCN-3 aportan la información necesaria para generar sistemas de prueba basados en la arquitectura

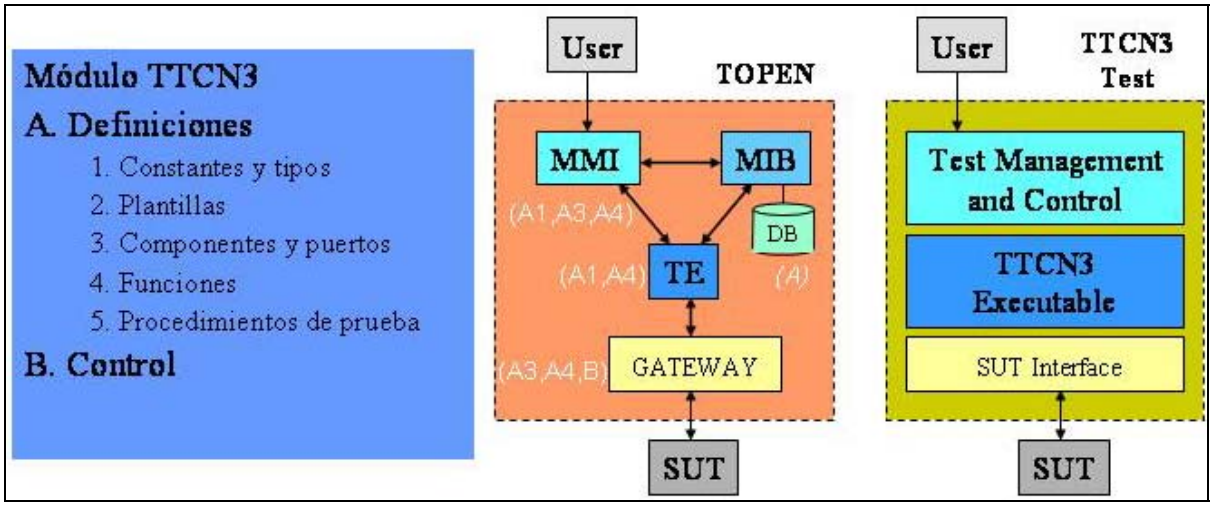

Figura 3. Generación de Sistemas de Pruebas.

TOPEN, tal y como describe a continuación:

- TE. La generación automática de este componente se fundamenta en la declaración de tipos de datos (etiqueta A1 en la figura 3) y funciones del módulo TTCN-3 (etiqueta A4). Por un lado permiten completar la gramática del lenguaje de operación/pruebas con los comandos de operación propios del SUT. Por otro, expresar el comportamiento de los comandos que se describe en las funciones del módulo: una función indica los puertos usados en la conexión y los datos enviados al SUT.
- MMI. Este componente requiere de los cambios pertinentes en el fichero externo de configuración, relativos a los comandos de operación y elementos que

comprende el SUT (etiquetas A1, A3 y A4). Estos cambios pueden realizarse automáticamente con parte de la información necesaria para generar el componente TE (sintaxis de los comandos de operación y componentes del SUT). Las plantillas de una especificación TTCN-3 (etiqueta A2) corresponden a los diferentes procedimientos de prueba que un usuario de TOPEN puede realizar desde el MMI.

- MIB. El modelo de datos es independiente del dominio de aplicación, por lo que este componente no requiere ninguna acción. Sin embargo, la base de datos se debe poblar con la información suministrada por las plantillas y procedimientos de prueba definidos en el módulo (etiquetas del grupo A). También la información de componentes y puertos permiten registrar la configuración de los elementos que componen el SUT.
- Gateway. Este componente que hace de enlace entre TOPEN y el SUT, puede generarse a partir de las definiciones de componentes y puertos (etiqueta A3), del comportamiento de las funciones (etiqueta A4) y de la parte de control del módulo (etiqueta B).

En cuanto al entorno de ejecución TTCN-3, desde el punto de vista arquitectónico, es posible realizar algo que se puede entender como la emulación con TOPEN (partes central y derecha de la figura 3). El componente TCM (Test Managment and Control) realiza tareas de gestión de componentes de pruebas, de gestión de entrada y análisis de datos) y está íntimamente ligado al componente MMI y MIB de TOPEN. El componente TTCN-3 Executable, cuya misión es la interpretación y ejecución de procedimientos de prueba, tiene su homólogo TOPEN en el componente TE donde los comandos son analizados para la búsqueda de errores sintácticos y semánticos y lanzada su ejecución mediante el Gateway. Finalmente, el componente System Under Test Interface, que implementa la interfaz de comunicación TTCN-3 con el SUT, tiene su homólogo en el componente Gateway de TOPEN (como se muestra en la figura 2).

5. Conclusiones

TTCN-3 es un lenguaje poderoso para definir procedimientos de prueba junto con la infraestructura de pruebas requerida para instrumentar la ejecución de las pruebas. Sin embargo no aborda el hecho de que el sistema SUT pueda ser operado a través de una

interfaz gráfica o una interfaz con una base de datos que permita realizar la gestión y análisis posterior de los resultados de las pruebas. Además, en TTCN-3, no se puede separar el proceso de construcción del sistema de pruebas del proceso de pruebas en sí. Por otra parte TTCN-3 es complejo lo que al final se puede convertir en una barrera para su uso. De hecho la inversión necesaria para utilizarlo adecuadamente puede ser grande y sólo compensar en casos en que toda la estrategia de un laboratorio en el que haya que probar diferentes SUTs se oriente a ella.

En este artículo se ha descrito una aproximación a la generación de un sistema de pruebas a partir de especificaciones en TTCN-3 apoyada en la arquitectura del entorno de pruebas TOPEN. El sistema de pruebas generado de esta forma permite centrarse en la definición de las pruebas en sí, al no tener que ocuparse de aspectos de implementación del sistema de pruebas. Por otra parte permite trabajar con una base de datos de resultados de las pruebas. La definición de un módulo TTCN-3 y el entorno de ejecución TTCN-3 nos aportan los elementos necesarios para plantear la automatización de las pruebas utilizando TOPEN. Por estas razones TTCN-3 puede llegar a ser una base de partida para probar sistemas utilizando TOPEN. En estos momento se está estudiando la conveniencia de soportar algunos aspectos concretos de TTCN-3 dentro de la filosofía que supuso el diseño de TOPEN de forma que se puedan conjugar las ventajas de ambos enfoques.

Agradecimientos

Este trabajo ha sido financiado por el Ministerio de Educación y Ciencia dentro del Plan Nacional de I+D+I, Proyectos TIC2003-08503 (AGMOD) y TIN2005-24792-E (REPRIS).

Referencias

[1] Object Management Group, "UML 2.0 Testing Profile Specification". OMG Final Adopted Specification, document reference ptc/2004-04-02. www.omg.org, 2004.

[2] J. Zander, Z.R. Dai, I. Schieferdecker, G. Din. "From U2TP Models to Executable Tests with TTCN-3 – An Approach to Model Driven Testing", IFIP 17th Intern. Conf. on Testing Communicating Systems – TestCom 2005. Montreal, Canada, March 2005.

[3] ETSI European Standard (ES) 201 873-1 version 2.2.1 (2003-02): The Testing and Test Control Notation version 3 (TTCN-3); Part1: TTCN-3 Core Language. 2003.

[4] TTCN-3 Homepage http://www.ttcn-3.org/

[5] T. Vassiliou-Gioles. "How to use the TTCN-3 Runtime and Control Interfaces TRI and TCI". TTCN User Conference June 2005. Sophia Antipolis, France.

[6] C. Willcock. "An Introduction to TTCN-3". TTCN User Conference June 2005.

[7] T. Vassiliou-Gioles. "The TTCN-3 Language". TTCN User Conference June 2006. Berlin, Germany
[8] Pedro P. Alarcón, Juan Garbajosa, Alberto Crespo, Belén Magro. "Automated integrated support for requirements-area and validation processes related to system development". In 2nd IEEE International Conference on Industrial Informatics INDIN´04. ISBN-0-7803-8513-6, IEEE Press, 2004.
[9] J. Garbajosa, M. Alandes, M.A. Mahillo, M. Piattini, "Assisting the Definition and Execution of Test Suites for Complex Systems". In Proceedings of the 7th International Conference on Engineering of Computer Based Systems, ECBS2000, pages 327-333, ISBN 0-7695-0604-6. IEEE Computer Society, 2000.
[10] B. Magro. Reference Architecture for a Validation Environment Family. PhD dissertation, Universidad Politécnica de Madrid. 2005.
[11] Pedro P. Alarcón, J. Garbajosa, A. Yagüe, J. Díaz. "TOPEN data model and analysis for systems validation". In 4th Workshop on System Testing and Validation. Postdam, Germany. March 2006.
[12] Cesar Viho, Aniie Floch, Frank Le Gall, Janie Baños, Carlos Pérez "The Go4IT Projectc: Toward a TTCN-3 open environment for IPv6 protocols testing" TTCN-3 User Conference 2006.
[13] Pulei Xiong, Robert L. Probert, Bernard Stepien. "An Efficient Formal Testing Approach for Web Service with TTCN-3". In TTCN-3 User Conference 2006.
[14] OpenTTCN Tester. http://www.openttcn.com, noviembre 2006
[15] Tau Tester. http://www.telelogic.com, noviembre 2006
[16] TTCN-3 toolbox. http://www.danet.com/
[17] TTworkbench. http://www.testingtech.de/products/ttwb_intro.php
[18] R. Meredith, "T3Doc Modifications & Updates Part 1: Bugs & Errors", March 2006.
[19] J.M. Lázaro, M.A. Mahillo, L.F. Peñafiel, A. Gonzalo, "Testing of Interconnected Machines". In 1st Workshop on System Testing and Validation. Paris, December 2002.

TOP 10 de factores que obstaculizan la mejora de los procesos de verificación y validación en organizaciones intensivas en software

Javier García, Antonio de Amescua, Manuel Velasco
Departamento de Informática
Escuela Politécnica Superior
Universidad Carlos III de Madrid
Avda. de la Universidad, 30, 28911 Leganés (Madrid)
{jgarciag, amescua}@inf.uc3m.es, velasco@ia.uc3m.es

Abstract

In spite of the potential benefits that the correct application of software verification and validation processes, techniques and tools can provide; the institutionalized use of them in the software industry does not reach the minimum capability levels required to gain the mentioned benefits. Moreover this circumstance is aggravated in the small and medium software intensive organizations due to the lack of available human and economic resources.
In this paper, the 10 more important factors that prevent the correct institutionalization of the software verification and validation efficient practices are presented. These factors are obtained from the authors' experience in several software process improvement initiatives related to software verification and validation processes.

Resumen

A pesar de los beneficios que se pueden obtener de la correcta aplicación de los procesos, técnicas y herramientas de verificación y validación de software, la utilización institucionalizada de los mismos en la industria, sobre todo en las pequeñas y medianas organizaciones de desarrollo de software, no alcanza los niveles mínimos para el logro de los beneficios potenciales.
En este artículo se presentan los 10 factores más importantes que impiden la correcta aplicación de las prácticas eficientes de verificación y validación de software, a partir de la experiencia recogida en los numerosos programas de mejora de procesos software en los que han participado los autores.

Palabras clave: Verificación, Validación, Pruebas de Software, Gestión del Cambio, Mejora de Procesos

1. Introducción

Se ha reportado en numerosos estudios [1] relativos al beneficio obtenido a raíz de procesos de mejora que la implantación de un proceso formal de verificación y validación supone importantes beneficios para las organizaciones software:

- Se produce un incremento de la satisfacción del cliente al utilizar un software con una cantidad de errores inferior.
- Se incrementa la eficiencia del proceso de desarrollo.
- Se facilita la definición y cumplimiento de los objetivos de calidad.
- Se incrementa la satisfacción de los trabajadores debido a que se proporcionan herramientas y recursos apropiados para la realización eficiente del trabajo.

Además, las empresas participantes en los mencionados estudios han reportado la consecución de beneficios económicos asignables a las actividades de mejora del proceso de verificación y validación de software son los siguientes:

- Reducción de un 20% en los errores en el software entregados al cliente (BKIN Software).
- El esfuerzo en pruebas de software se redujo desde el 25% al 20% del esfuerzo total del proyecto (BKIN Software).
- El número de errores detectados en las pruebas de aceptación es menor del 12% de los errores detectados en las pruebas de integración. El valor anterior era del 37%. (BKIN Software).
- Los errores informados por el cliente/usuarios se han reducido en un 77% (Archetypon).
- 30% de reducción en los costes de ejecución (IMB SEMEA SUD).
- Reducción en el tiempo de entrega e incremento en la eficiencia de las pruebas (Nokia – Network Management Systems).

Figura 1. Beneficios en la mejora de los procesos de verificación y validación software

Sin embargo, a pesar de los beneficios que una aplicación institucionalizada de procesos de verificación y validación de software, según el perfil de madurez mundial de las organizaciones intensivas en software [2] elaborado por el Software Engineering Institute (SEI), de aquellas organizaciones (402 en 2006) que evaluaron sus procesos con respecto al nivel 2 de capacidad establecido por el CMMI para el Proceso de Verificación, solo el 2,98% de las organizaciones (12) lograron satisfacer totalmente ese nivel, mientras que el 7,96% organizaciones (32) lo lograron satisfacer parcialmente es decir, no en la totalidad de los requisitos marcados por el CMMI).

Por otra parte, para aquellas organizaciones (45 en lo que llevamos de 2006) que evaluaron sus procesos con respecto al nivel 1 de capacidad establecido por el CMMI para el Proceso de Verificación, solo el 8,88% de las organizaciones (4) lograron satisfacer totalmente ese nivel, mientras que el 40% de las organizaciones (18) lo lograron satisfacer parcialmente.

Ante estas cifras tan bajas, es necesario que se analicen los factores que impiden que las prácticas eficientes relativas a la verificación y validación se introduzcan, asimilen y apliquen de manera institucionalizada y efectiva en organizaciones software de todo tipo.

2. Modelo de referencia para clasificar los factores que obstaculizan la mejora de los procesos de verificación y validación de software

El modelo IDEAL [3], como otras aproximaciones para la introducción de mejoras en el proceso productivo de una organización (i.e. Plan-Do-Check-Act, PDCA), tiene como propósito establecer los mecanismos para facilitar la realización de programas de mejora continua en organizaciones, aunque el modelo IDEAL está especialmente orientado a las mejoras en organizaciones intensivas en software. Estos modelos de mejora definen las siguientes fases para la realización de un programa de mejora de procesos:

1. **Obtención del compromiso**, cuyo propósito consiste en establecer los objetivos que se deberán alcanzar con la utilización sistemática de la mejora, en este caso en el proceso de verificación y validación, desarrollar el plan para la realización de las mejoras y la obtención del compromiso requerido en cuanto a los objetivos, actividades, calendario y recursos disponibles para el programa de mejora.

2. **Diagnosis**, cuyo objetivo consiste en establecer las prácticas eficientes ya existentes en la organización e identificar las necesidades y oportunidades concretas, en este caso, relativas a la mejora de los procesos de verificación y validación.
3. **Definición**, que persigue la definición del proceso y de todas las guías que permitan adaptar el proceso general definido a cada uno de los tipos de trabajos realizados por la organización.
4. **Implantación**, cuyo propósito es el logro de la utilización generalizada en todos los trabajos de la organización del proceso mejorado.
5. Por ultimo, la fase de **análisis de resultados** tiene como propósito cuantificar las mejoras logradas con la introducción del nuevo proceso y determinar los próximos objetivos de mejora a satisfacer.

Por otra parte, para poder aplicar convenientemente cualquiera de los modelos de mejora es necesario que se aplique un modelo de referencia que establezca los objetivos de los procesos de verificación y validación, así como las actividades y tareas que permiten conseguir eficientemente el propósito de los mismos.

A lo largo de varios programas de mejora de los procesos de verificación y validación de software, se han detectado diferentes dificultades que se repiten en todos ellos. Estas dificultades se han clasificado según la fase del modelo IDEAL en la cual se manifiestan. Para cada uno de los problemas reflejados se presenta su enunciado, su descripción detallada y la solución que los autores de la ponencia han aplicado con éxito.

3. Descripción de las organizaciones de desarrollo de software consideradas en el ámbito de este trabajo

Los factores para la mejora de los procesos de verificación y validación en organizaciones intensivas en software que se presentan en este trabajo, se han identificado a partir de las lecciones aprendidas que los autores han recopilado a lo largo de las 10 actividades de mejora de procesos de software en las que han participado. En la tabla 1 (anexo 1), se muestran las características que identifican la tipología de las organizaciones consideradas en el ámbito de este trabajo. Debido a restricciones contractuales, los autores de este trabajo no están autorizados a divulgar los nombres de las organizaciones participantes en las mencionadas actividades de mejora de procesos software.

4. Lista de factores que obstaculizan la mejora de los procesos de verificación y validación de software

A continuación se presentan los factores identificados, teniendo en cuenta en que éstos NO están ordenados por orden de importancia o relevancia.

4.1 Factores relativos a los modelos de referencia

En la actualidad existen distintos modelos de referencia (ISO 12207 [5], CMMI [6] e IEE1074 [7] entre otros) que describen los procesos de verificación y validación de sistemas informáticos (y por inclusión, aquellas actividades relativas a verificación y validación de software), sus objetivos y prácticas eficientes. Sin embargo, existen distintos problemas que dificultan su uso para determinar las prácticas eficientes, necesidades y oportunidades de mejora durante la fase de diagnosis de la mejora, así como su uso como guía para la mejora en la organización.

FACTOR 1	
Enunciado	**Los modelos de referencia que definen las prácticas de verificación y validación no son fáciles de usar**
Descripción	Todo el mundo entiende que los modelos de referencia son difíciles de entender porque: ▪ Utilizan terminología que, en numerosos casos, no es conocida por el personal de las organizaciones. ▪ Asimismo, la terminología utilizada en la organización no es conocida por los consultores que ayudan a la organización en la mejora de los procesos de verificación y validación. ▪ Son documentos muy densos que suelen comprimir en muy pocas páginas gran cantidad de conceptos en frases condensadas, lo que dificulta su comprensión.
Solución aplicada	Los equipos de mejora deben estar conformados por personas que tengan distintos perfiles: ▪ Expertos en el modelo de referencia de procesos a implantar, en técnicas de revisión y en pruebas. ▪ Jefes de Proyecto con visión global de todos los tipos de trabajo de la organización. ▪ Personal con gran experiencia y bagaje en el diseño, construcción, prueba y gestión de los proyectos de la organización.
FACTOR 2	
Enunciado	**Los modelos de referencia no proporcionan la cohesión necesaria entre los procesos de verificación y validación con el resto de procesos de ingeniería[1] necesarios para desarrollar software**
Descripción	Los modelos de referencia considerados realizan una descripción detallada de las actividades de cada proceso de verificación y validación. Sin embargo, es conocido que las actividades de estos procesos deben ejecutarse sincronizadamente con otras actividades de ingeniería y gestión propias del proyecto. Estas interacciones: ▪ No se describen en detalle en los modelos de referencia de procesos software. ▪ En caso de que se describan, solamente se realizan mediante la inclusión de una referencia al proceso con el que se debe interactuar, sin indicar cómo se deben sincronizar ni la información que debe ser proporcionada/usada por cada uno de los procesos sincronizados. Por tanto, es necesario realizar un trabajo adicional que permita establecer colaboraciones eficientes entre las actividades de los procesos de verificación y validación con las actividades de ingeniería.

[1] Se utiliza la expresión "procesos de ingeniería" para hacer referencia a los procesos de especificación de requisitos, análisis, diseño, construcción, despliegue y mantenimiento de sistemas de información.

Solución aplicada	A lo largo de los programas de mejora realizados en las organizaciones indicadas en la sección 3, los autores han desarrollado dos tipos de soluciones: ▪ Mapas de los modelos de referencia para los procesos de verificación y validación, que enriquecen la información proporcionada por los modelos de referencia con: perfiles que deben interactuar y para cada práctica o actividad que requiere la interacción con cada proceso, se especifica la información que se debe proporcionar o utilizar. ▪ Definición de guía por perfiles, en las que para cada uno de los perfiles que intervienen, se establecen las actividades de verificación y validación y de otros procesos estableciendo el flujo de trabajo normal desde que se comienza un trabajo hasta que se finaliza.

Tabla 1. Problemas relativos a los modelos de referencia.

4.2. Obtención del compromiso para la definición o mejora del proceso de verificación y validación

Una de las principales dificultades de cualquier proceso de mejora consiste en la obtención del soporte y patrocinio necesario para la organización, así como el logro de los compromisos necesarios entre los objetivos de mejora y los recursos y plazos disponibles. Estos factores se ven incrementados cuando se intenta mejorar cualquier proceso relacionado con las pruebas, los más destacados se muestran en la tabla 3.

FACTOR 3	
Enunciado	**Es difícil estimar el ROI esperado de la mejora en el proceso de verificación y validación y las estimaciones obtenidas suelen ser poco fiables**
Descripción	Cuando se evalúa la conveniencia, necesidad y oportunidad de emprender un programa de mejora, la alta dirección necesita conocer cual será el retorno previsible de la inversión que se realice. Para ello, es necesario estimar los costes del programa de mejora (que pueden ser estimados a priori por expertos con poco margen de error [8]) y también hay que cuantificar los resultados obtenidos, tarea para la cual aún no existe ningún modelo experimentado con éxito.
Solución aplicada	Para estimar el coste de la calidad y la no calidad, en algunos proyectos de mejora, los autores han aplicado el siguiente modelo lineal computado a partir de los siguientes elementos: ▪ El coste asociado a cada uno de los errores o no conformidades detectados en el proyecto y del retraso con respecto al inicio con se producen, de tal manera que se asigna un multiplicador de esfuerzo distinto a cada error que aumenta según avanza el tiempo desde el inicio del proyecto. Además este multiplicador ha sido distinto en cada una de las organizaciones en las que se ha aplicado esta solución. ▪ El coste requerido para hacer todas las revisiones y pruebas que ha sido registrado en los ficheros de esfuerzo de la organización. Este modelo está aún en investigación y se ha aplicado con desigual resultado. Asimismo, se está comenzando a trabajar en la compilación y estandarización de un conjunto de casos de estudio y en un modelo que permita la comparación de estos casos de estudio con las características de una organización que pretenda mejorar el proceso de verificación y validación.
FACTOR 4	
Enunciado	**El coste estimado de las acciones de mejora del proceso de verificación y validación es tan elevado que la organización (sobre todo las pequeñas y medianas) no se lo pueden permitir, aunque los beneficios pudieran llegar a ser muy espectaculares**
Descripción	El coste para la mejora de los procesos de verificación suele incluir el valor económico del tiempo que el personal de la organización dedica a la mejora, el coste del equipo de consultores externos que suele participar para guiar en el proceso de mejora, así como el coste asociado a las nuevas herramientas que se introducen en la organización para automatizar los procesos de verificación y validación definido. La inversión inicial puede llegar a ser tan elevada que la organización que deba hacerla

	no pueda permitírsela, de tal manera que el potencial beneficio aunque pueda llegar a ser muy alto no podrá ser alcanzado por la organización.
Solución aplicada	La solución propuesta por los autores está basada en: ▪ La definición de patrones [9] de verificación y validación. Estos patrones, desarrollados por los autores de la referencia citada, son soluciones prefabricadas para aplicar los mencionados procesos bajo distintos tipos de desarrollo. Estos patrones proporcionan información relativa a: Nombre, Patrones relacionados, Contexto inicial, Contexto de resultado, Problema, Solución, Entradas y Salidas. ▪ Recursos formativos y guías electrónicas que proporcionan instrucciones técnicas acerca de cómo utilizar un patrón de verificación y validación con un conjunto de herramientas específicas para el control de defectos, revisiones y pruebas. ▪ Modelos de negocio que permiten la publicación y adquisición de patrones y guías electrónicas, de tal manera que varias organizaciones puedan compartir el uso y el pago de los costes de los mismos. Estas soluciones reducen la inversión inicial requerida, permitiendo a pequeñas y medianas organizaciones de desarrollo software compartir los costes para la mejora de los procesos software y, en concreto, los relativos a la verificación y validación.

Tabla 2. Problemas relativos a la obtención del patrocinio de la dirección.

4.3. Diagnosis de la situación actual

Una vez que se han conseguido los propósitos de la fase de establecimiento de compromiso, es necesario determinar la situación actual que suele ser considerada muy negativamente por el personal de una organización que participa por primera vez en un programa de mejora. Este problema se describe en detalle en la tabla 4.

FACTOR 5	
Enunciado	**La fase de diagnosis de la situación actual del proceso de verificación y validación (incluyendo las pruebas) es una pérdida de tiempo y dinero, ya sabemos que no tenemos un proceso común**
Descripción	Normalmente la fase de diagnosis está dirigida a detectar las carencias de las prácticas actuales de verificación y validación realizadas por la organización e identificar los aspectos que deben ser mejorados. Esta información no es reconocida como valiosa como numerosos miembros de la organización, lo que puede suponer un rechazo acerca del programa de mejora.
Solución aplicada	La solución planteada por los autores se centra en que el producto final de la fase de diagnosis es la definición del proceso actual identificando aquellas prácticas que ya son ejecutadas por una minoría, se pueden generalizar para mejorar el desempeño de la organización. Este plan de acción está compuesto de los siguientes tipos de actividades: ▪ Creación de un patrón de procesos ▪ Adaptación de un patrón de procesos existente ▪ Creación de plantillas con la estructura de los productos generados / utilizados por un patrón ▪ Creación de una instrucción técnica para un patrón ▪ Adquisición o creación de ejemplos de aplicación de patrones y de los productos generados por los mismos. ▪ Realización de actividades de soporte para ayudar al personal a aprender a usar los activos (patrones, guías y plantillas) definidos. ▪ Realización de actividades de monitorización de la utilización de los activos definidos. ▪ Dotación de la infraestructura requerida para implantar las mejoras definidas. Este plan de acción, además de una reducción de trabajo en futuras actividades, supone un aumento de la motivación del personal de la organización aumentando su implicación en las actividades de mejora, factor crítico de éxito para la consecución de sus objetivos.

Tabla 3. Problemas relativos a la determinación de la situación actual.

4.4. Definición de procesos eficientes y optimizados

La definición del nuevo proceso de verificación y validación, así como su adaptación es una de las fases más problemáticas de la mejora por las discusiones y conflictos que se producen a lo largo de las tareas en grupo. Los principales problemas que suelen aparecer durante este trabajo se describen en la tabla 5.

FACTOR 6	
Enunciado	**¿Qué es primero: el nuevo proceso de pruebas o las herramientas que permiten su automatización?**
Descripción	En muchas organizaciones, la mejora de las actividades para la prueba de software se produce como consecuencia a la adquisición de una herramienta para la automatización de este tipo de actividades, lo que supone que las mejoras del proceso se restrinjan única y exclusivamente a las prestaciones proporcionadas por la herramienta en cuestión, lo que suele limitar la capacidad de mejora.
Solución aplicada	La primera actividad de la definición del nuevo proceso de verificación y validación debe ser identificar la configuración de actividades. En ese momento, antes de definir el procedimiento detallado, se debe producir la selección de la herramienta que permita automatizar las actividades definidas con mayor eficacia y ahorro de costes. Posteriormente, y en colaboración con los expertos en la herramienta, se elaborarán los procedimientos e instrucciones para la ejecución eficiente, con la herramienta, de las actividades del proceso de verificación y validación. Es necesario indicar que las herramientas que se han introducido en primer lugar en todas las organizaciones estudiadas están dirigidas a la gestión de los defectos detectados durante las revisiones y pruebas de software. Asimismo, en casi todas las organizaciones se han implantado herramientas de uso libre para la automatización de las pruebas unitarias (NUnit y JUnit). En ninguna de las organizaciones se ha adquirido a nivel organizativo herramientas específicas para la prueba de rendimiento y carga de un sistema o para la automatización de las pruebas de aceptación de un sistema.
FACTOR 7	
Enunciado	**El establecimiento de revisiones técnicas, por pares o mediante inspecciones formales supone una sobrecarga de trabajo que no podemos asumir**
Descripción	Personal participante en numerosas actividades de mejora de proceso, cuando se plantea la inclusión de actividades y técnicas para la realización de revisiones técnicas, por pares o mediante inspecciones formales, de los productos de análisis y diseño, aducen que este tipo de tareas es muy eficaz desde el punto de vista teórico, pero, en la práctica, producen un sobrecoste y una sobrecarga de trabajo la organización no puede asumir.
Solución aplicada	Las soluciones aplicadas por parte de los autores en las organizaciones consideradas se centran en los siguientes aspectos: ▪ Se han proporcionado listas de verificación adaptadas a los tipos de software elaborados y a las actividades realizadas (nuevos proyectos, mantenimientos adaptativos, correctivos, perfectivos o preventivos), de tal manera que los revisores tienen unas guías que les ayudan en la realización de las revisiones y en la detección de defectos más comunes. ▪ Introducción de métricas, basadas en los datos proporcionados por las listas de verificación, que permitan mostrar a los equipos de desarrollo la efectividad de las revisiones. Entre las métricas utilizadas son: tasa de errores detectados por cada revisión, esfuerzo medio requerido en realizar cada tipo de revisión y estudio de la evolución de las correcciones solicitadas por los usuarios para los productos entregados antes y después de aplicar las revisiones técnicas. ▪ Motivación mediante la proyección del coste de la calidad y la no calidad utilizando el modelo experimental presentado en la descripción de la solución del problema 4 e introducción de actividades de revisión con una exigencia menor de la indicada por los modelos de referencia, que se incrementa en sucesivas acciones de mejora.

FACTOR 8	
Enunciado	**La definición de los casos de prueba de sistema es una actividad innecesaria y burocrática**
Descripción	La definición detallada de los casos de prueba de sistema, indicando todos los pasos se suele considerar una tarea burocrática que requiere mucho esfuerzo y que no tiene un beneficio claro, porque en muchos casos, la persona que escribe los casos es el que ejecuta la prueba.
Solución aplicada	La solución propuesta tiene una doble vertiente: ▪ Por un lado, los autores han aplicado con éxito plantillas de casos de pruebas con campos precargados en los que los pasos se pueden especificar mediante un seudocódigo (lenguaje común) conocido por los miembros de la organización. Asimismo, se han proporcionado ejemplos de casos de prueba que se depositaban en una carpeta compartida. Los contenidos de esta carpeta compartida se enriquecen paulatinamente mediante la incorporación de los casos de prueba elaborados por los equipos de desarrollo una vez introducida la técnica. En todas las organizaciones, se ha determinado necesario, proporcionar mecanismos de búsqueda que faciliten la localización de aquellos casos de prueba que pueden ser reutilizados en otros trabajos. ▪ Por otra parte, es necesario cuantificar la reducción del coste de la construcción de los casos en las pruebas de regresión consecuencia de un mantenimiento utilizando una especificación que exista previamente, lo cual se puede conseguir con un pequeño experimento. Además, dependiendo de las características organizativas, se pueden aplicar iniciativas de rotación de personal, siempre y cuando se disponga de personal capacitado para desempeñar distintos roles en un proyecto de desarrollo de software

Tabla 4. Problemas relativos a la definición del nuevo proceso de verificación y validación.

4.5. Implantación de los procesos mejorados

Los riesgos no se han acabado una vez que se ha logrado definir un proceso de verificación y validación que sea eficiente y cumpla las expectativas de la organización, es necesario capacitar a los ingenieros del software usuarios del proceso. Por otra parte, también es necesario lograr el soporte por parte de todos los afectados de que las actividades de verificación y validación no sean las primeras sacrificadas una vez que se producen retrasos en el proyecto. Estos problemas se describen en detalle en la tabla 6.

FACTOR 9	
Enunciado	**Es escasa la formación práctica de los ingenieros de software acerca de las actividades de revisión, inspección y pruebas de software**
Descripción	Uno de las principales dificultades de la implantación eficaz de un nuevo proceso de verificación y validación consiste en que muchos de sus usuarios no conocen o recuerdan los conocimientos teóricos imprescindibles para realizar una eficaz revisión del contenido de un análisis o diseño o una prueba fiable de una función, clase, módulo o componente software.
Solución aplicada	La formación previa a la implantación de la mejora no debe ceñirse al aprendizaje de la utilización de una herramienta de pruebas de software concreta, sino que debe incluir recordatorios o sesiones de técnicas de pruebas estructurales y funcionales, así como la realización práctica de inspecciones formales o revisiones por pares (dependiendo de las técnicas que se implanten en cada organización). Además, existen programas continuados de formación con la siguiente estructura: ▪ En primer lugar, se ha elaborado un índice de las habilidades requeridas para la aplicación efectiva de los procesos de verificación y validación. ▪ Posteriormente, el personal ha asistido a sesiones de formación intensiva (≈ 20 – 30 horas) para conocer los procesos de verificación y validación definidos, así como los fundamentos de las técnicas básicas requeridas. ▪ Por último, se ha acordado celebrar sesiones de refresco o recordatorio para reforzar aspectos más controvertidos o difíciles en los procedimientos definidos.

FACTOR 10	
Enunciado	**El tiempo de retraso de un proyecto es directamente proporcional a la disminución de tiempo y esfuerzo dedicado a la actividades de verificación y validación y, más específicamente, a las actividades de prueba de software**
Descripción	Normalmente, cuando se retrasan los plazos de tiempo la dirección de la organización opta por aumentar el tiempo de desarrollo y recortar el tiempo de pruebas, asumiendo los posibles costes asociados a la realización de mantenimientos correctivos posteriores. Este fenómeno, a parte de tener un impacto negativo en la imagen de la organización, también mina la motivación del personal de la organización para aplicar sistemáticamente las mejoras introducidas.
Solución aplicada	La única solución a este problema reside en que la dirección de la organización manifieste su compromiso con la entrega de productos de calidad y establezca las acciones correctivas necesarias para la correcta verificación y validación de todo aquel software que se entregue, renegociando los compromisos en cuanto a plazos o alcance de la entrega.

Tabla 5. Problemas relativos a la obtención del patrocinio de la dirección.

5. Conclusiones

En este artículo se han presentado los beneficios que la mejora en el proceso de verificación y validación (y por tanto de pruebas de software) puede proporcionar para una organización intensiva en software, pero que, mundialmente, son pocas las organizaciones que han logrado la capacidad suficiente para que se logren en toda su expresión los beneficios esperados de estos procesos.

Asimismo, se han enumerado 10 factores más relevantes que impiden el logro de la madurez requerida, estableciendo posibles soluciones a los mismos basadas en la experiencia práctica de los autores de este trabajo en experiencias de mejora de los procesos de verificación y validación en organizaciones intensivas en software. Además, estas soluciones suponen líneas de investigación que están comenzando a ser abordadas por el grupo de Ingeniería del Software del Departamento de Informática de la Universidad Carlos III de Madrid. Estas líneas están relacionadas con:

- Métodos de estimación para el cálculo del ROI de la mejora de los procesos de verificación y validación.
- Métodos de estimación de los costes relativos a la mejora de los procesos software y, en concreto, los relativos a la verificación y validación.
- Definición de recursos y modelos de negocio que permitan a pequeñas y medianas organizaciones de desarrollo software compartir los costes para la mejora de los procesos software y, en concreto, los relativos a la verificación y validación.
- Definición de modelos cuantitativos que permitan calcular el valor añadido y el beneficio que cada acción de mejora de los procesos de verificación y validación supone sobre los resultados obtenidos por la organización.

Agradecimientos

Este trabajo ha sido apoyado por el Ministerio de Educación y Ciencia dentro del proyecto TIN2005-24792-E (REPRIS).

Referencias

[1] SPIRE Cases Studies. *Software Process Improvement in Regions of Europe.* ESSI project 23873 SPIRE. (http://www.cse.dcu.ie/spire/main.html)

[2] *CMMI Maturity Profile March 2006 Report.* Software Engineering Institute. Carnegie Mellon University, March 2006 (http://www.sei.cmu.edu/appraisal-program/profile/pdf/CMMI/2006marCMMI.pdf)

[3] B. McFeeley, *IDEAL: A User's Guide for Software Process Improvement, tech. report* CMU/SEI-96-HB-001, Software Eng. Inst., 1996.

[4] Deming, WE *The New Economics: for industry, government, education*. 1994 MIT CAES, Cambridge

[5] Paulk, M.C.; Curtis, B.; Chrissis, M.B.; Weber, C.V. *CMMI^{SM} Product Suite*, www.sei.cmu.edu/cmmi/products/products.html, Sept. 2001.

[6] ISO 12207.0-1995, Industry Implementation of International Standard ISO/IEC 12207: 1995 (ISO/IEC 12207) *Standard for Information Technology, Software life cycle processes IEEE/EIA 12207.0-1996* (A Joint Standard Developed by IEEE and EIA),March 1998.

[7] IEEE Std 1074-1997. *IEEE Standard for Developing Software Life Clycle Processes*. Institute of Electronics Engineers. 1997.

[8] Rico, D.F., *ROI of Software Process Improvement: Metrics for Project Managers and Software Engineers*, J. Ross Publishing, Inc., February 2004

[9] Antonio Amescua, Javier García, Maria-Isabel Sánchez-Segura, Fuensanta Medina-Domínguez, *A pattern-based solution to bridge the gap between theory and practice in using process models*, International Software Process Workshop and International Workshop on Software Process Simulation and Modeling, SPW/ProSim 2006, Shanghai, China, May 20-21, 2006, Proceedings Series: Lecture Notes in Computer Science , Vol. 3966

ANEXO 1

	Unidad de Desarrollo 1	Unidad de Desarrollo 2	Unidad de Desarrollo 3	Unidad de Desarrollo 4	Unidad de Desarrollo 5	Unidad de Desarrollo 6	Unidad de Desarrollo 7	Unidad de Desarrollo 8	Unidad de Desarrollo 9	Unidad de Desarrollo 10
Tamaño	≈ 30	≈ 50	≈ 25	≈ 10	≈ 35	≈ 100	≈ 100	≈ 100	≈ 100	≈ 100
Tipo de software que desarrolla	Software de gestión en entorno web y cliente/ servidor en Windows	Software de gestión en entorno web y cliente/ servidor en Windows	Software integrado en dispositivos hardware para medición	Software para proyectos de investigación e innovación en las TIC	Software de gestión en entorno host e interfaces web para el mismo	Implantación y adaptación de sistemas ERP	Implantación de soluciones comerciales para gestión de cono-cimiento	Implantación y adaptación de sistemas comerciales para comercio electrónico	Adaptación de herramientas de soporte para desarrollo de software	Implantación y adaptación de sistemas comerciales para gestión de logística
Tipos de trabajos realizados	Desarrollo de nuevas soluciones y mantenimien-to de las mismas	Desarrollo de nuevas soluciones y mantenimien-to de las mismas	Desarrollo de nuevas soluciones y mantenimien-to de las mismas	Desarrollo de nuevos componentes software	Desarrollo de nuevas soluciones y mantenimien-to de las mismas	Adaptación de soluciones comerciales existentes	Adaptación de soluciones comerciales existentes	Adaptación de soluciones comerciales existentes	Adaptación de soluciones comerciales existentes	Adaptación de soluciones comerciales existentes
Tecnologías de base utilizadas	Windows, NET y SQLServer	Windows, PLSQL, Java y Oracle	Windows, Linux, Java, .NET	Windows, Linux, Java, .NET, RFID, SQLServer, Oracle, etc.	AS/400, PDM, Java, WASAD y DB2	Windows, Linux, Java, .NET, lenguajes propietarios	Windows, Linux, Java, .NET, lenguajes propietarios	Windows, Linux, Java, .NET, lenguajes propietarios	Windows, Linux, Java, .NET, lenguajes propietarios	Windows, Linux, Java, .NET, lenguajes propietarios
Herramientas de pruebas usadas	Herramientas de gestión de defectos propia y NUnit	Herramientas de gestión de defectos propia y JUnit	JUnit y NUnit	JUnit, Nunit y herramientas de gestión de defectos para cada proyecto	Sin herramientas	JUnit y Nunit	JUnit y Nunit	JUnit y Nunit	JUnit y Nunit	JUnit y Nunit
Rotación de personal	Baja – Grupos de trabajo estables	Baja – Grupos de trabajo estables	Baja – Grupos de trabajo estables	Alta – Grupos de trabajo que cambian semestralmente	Muy Baja – Grupos de trabajo estables	Muy Alta – Distintos para cada trabajo	Muy Alta – Distintos para cada trabajo	Muy Alta – Distintos para cada trabajo	Muy Alta – Distintos para cada trabajo	Muy Alta – Distintos para cada trabajo
Objetivo para iniciar la actividad de mejora	Instauración de un proceso eficiente y guías de adaptación a los distintos trabajos	Instauración de un proceso eficiente y guías de adaptación a los distintos trabajos	Instauración de un proceso eficiente y guías de adaptación a los distintos trabajos	Instauración de un proceso eficiente y guías de adaptación a los distintos trabajos	Obtención de un certificado ISO 9001:2000 para desarrollo de software	Nivel 2 de capacidad según CMMI de procesos de desarrollo y gestión de proyectos	Nivel 2 de capacidad según CMMI de procesos de desarrollo y gestión de proyectos	Nivel 2 de capacidad según CMMI de procesos de desarrollo y gestión de proyectos	Nivel 2 de capacidad según CMMI de procesos de desarrollo y gestión de proyectos	Nivel 2 de capacidad según CMMI de procesos de desarrollo y gestión de proyectos

Tabla 6. Características de las organizaciones de las que se han obtenido experiencias de mejora de los procesos de verificación y validación.

Reseña sobre el taller de Pruebas en Ingeniería del Software 2006 (PRIS)

Pablo Javier Tuya González
Departamento de Informática
Universidad de Oviedo
tuya@uniovi.es

El Taller sobre Pruebas en Ingeniería del Software (PRIS 2006: http://in2test.lsi.uniovi.es/pris2006/) fue celebrado en Sitges el 3 de Octubre de 2006, en el marco de las Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2006). Dicho taller fue organizado como parte de las actividades de la Red para la promoción y mejora de las Pruebas en Ingeniería del Software (RePRIS: http://in2test.lsi.uniovi.es/repris/), financiada por el Plan Nacional de I+D+I del Ministerio de Educación y Ciencia y fondos FEDER (acción especial TIN2005-24792-E).

El objetivo general del taller fue establecer un foro de discusión sobre las actividades de I+D que se realizan en España relacionadas con la prueba del software. Esta primera edición del taller fue exitosa: contó con más de 20 asistentes y se presentaron 9 contribuciones diferentes seleccionadas mediante un proceso de revisión por pares.

Las contribuciones presentadas trataron tres temáticas diferentes. En primer lugar se presentaron y discutieron diferentes líneas de investigación realizadas por Departamentos Universitarios sobre pruebas tempranas, pruebas de servicios web y diagnosis. En el segundo bloque se trataron diversos estudios sobre la práctica y la investigación en pruebas sobre bases de datos, los principales problemas relativos a la mejora del proceso de verificación y validación, y la práctica profesional en España. El tercer bloque se centró en herramientas para soporte de las pruebas: pruebas unitarias, estándares TTCN-3 y soluciones comerciales de Telelogic.

Finalmente, hay que destacar la progresiva aparición y consolidación durante los últimos años de diversos grupos de investigación que abordan la temática de las pruebas desde diferentes puntos de vista. Para ellos este taller ha servido como medio de difusión de sus interesantes trabajos. Esta edición, y las futuras, contribuirán sin duda a elevar la calidad científica de la investigación en esta disciplina y a aportar una visión práctica que incida finalmente en la mejora de procesos de pruebas realizados en la industria.