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.

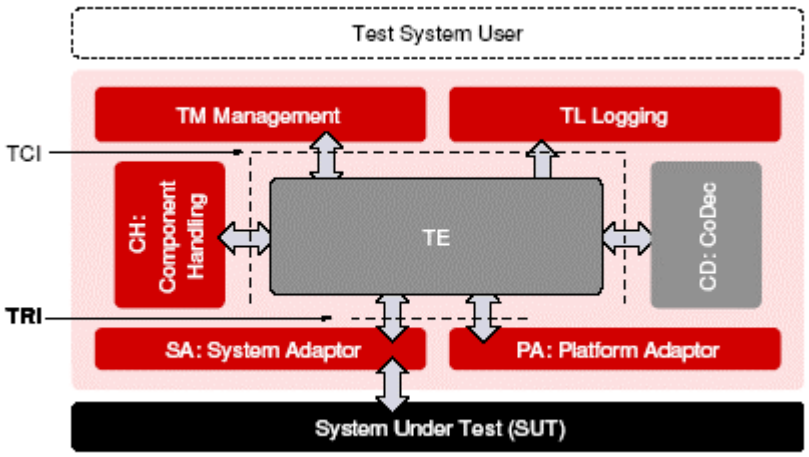

Figura 2. Arquitectura de un sistema de pruebas TTCN-3.

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

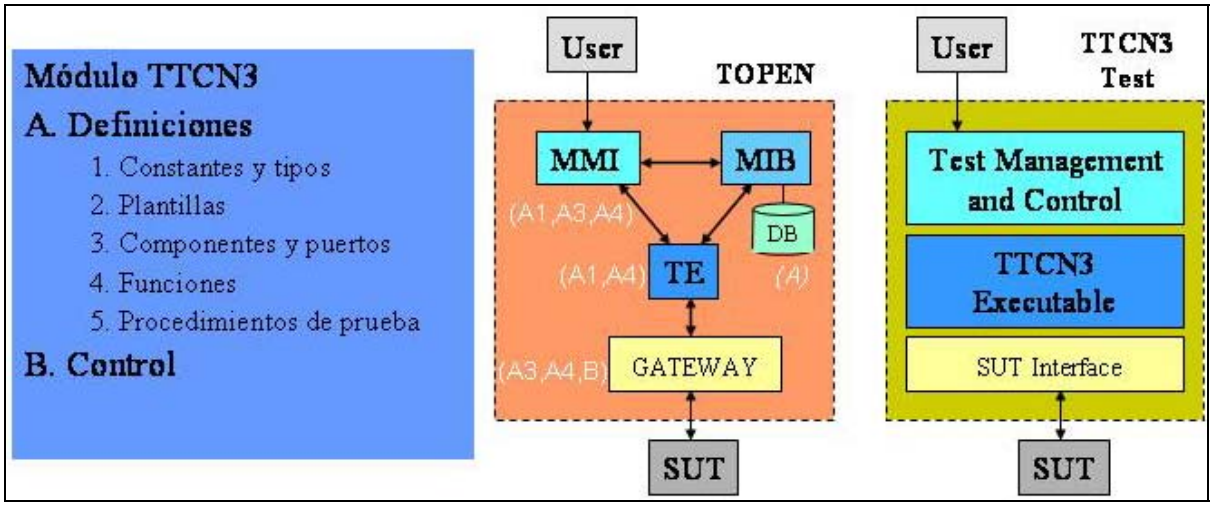

Figura 3. Generación de Sistemas de Pruebas.

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.