

## Revisión Sistemática: Pruebas de Software para SOA con UML

M<sup>a</sup> del Pilar Romay  
Dep. Ingeniería Sistemas  
Información y Tel.  
Universidad San Pablo CEU  
[pilar.romay@gmail.com](mailto:pilar.romay@gmail.com)

Luis Fernández-Sanz  
Dep. Ciencias de la  
Computación.  
Universidad de Alcalá  
[luis.fernandezs@uah.es](mailto:luis.fernandezs@uah.es)

Roberto Tejedor  
Meta 4  
[robertota@meta4.com](mailto:robertota@meta4.com)

### Resumen

El objetivo de este trabajo es realizar un estudio del estado actual de la investigación en pruebas de software para arquitecturas orientadas a servicio, considerando de manera específica el uso de UML como lenguaje de modelado. En primer lugar, se revisa el contexto general del estudio, detallando las cuestiones más cruciales referidas tanto a SOA y el concepto de servicio, como al uso de UML y sus extensiones como lenguajes de modelado, aplicados a las pruebas de software y a los propios servicios. A continuación se plantea una revisión sistemática siguiendo el método de Kitchenham, en la que se realizan búsquedas sobre UML, SOA y las pruebas de software, indicando tanto las preguntas como los criterios de exclusión usados. Se muestra que la mayoría de los resultados se agrupan en los campos del *Model-Based Testing* y el más específico *Model-Driven Testing*, a los que se dedican la mayoría de los artículos existentes; éstos son muy recientes, datándose los más antiguos de 2007. Se concluye que el campo de investigación es aún joven, y en él hay lugar para aportaciones significativas. También se concluye que UML no es específicamente muy utilizado en este contexto por sus necesidades particulares, aunque sí lo es de manera indirecta mediante la definición de *profiles* específicos.

### 1. Introducción

La aparición hace unos años de los sistemas software orientados a servicios, así como la gran expansión que han sufrido en la actualidad, hace muy importante cualquier avance en materia de definición de metodologías o de automatización de procesos de desarrollo de estos sistemas, entre los que se encuentran, con una importancia vital, sus procesos de aseguramiento de la calidad.

Este proceso de aseguramiento de la calidad incluye un complejo proceso de elaboración (y modelado) de las pruebas a realizar. Dado que los sistemas orientados a servicios son distintos, o tienen ciertas particularidades respecto a los sistemas más habituales (orientados a objetos), se considera necesario hacer un estudio de su estado, a día de hoy, con el objeto de comprobar a qué nivel están definidos los procesos, estándares y metodologías para este tipo de sistemas.

La complejidad a la que se está llegando dentro de los desarrollos software hace necesaria la utilización de modelos que representen y reduzcan esa complejidad: de ahí que se considere el uso de lenguajes de modelado específicos. El lenguaje más representativo en el proceso de desarrollo de software actual es el Lenguaje Unificado de Modelado (UML), actualmente en su versión 2.1 [1]. Sus capacidades de extensión le permiten crear lenguajes específicos de dominio (DSLs) a través de diversos *profiles* y extensiones del metamodelo. Esto hace posible usarlo como herramienta dentro del campo de pruebas.

Partiendo de las consideraciones anteriores, este artículo presenta una revisión sistemática del área de Pruebas de Software para Arquitecturas Orientadas a Servicios (SOA), utilizando UML como lenguaje de modelado. Esta revisión se centra principalmente en artículos de revista obtenidos a través de las llamadas Bases de Datos de Conocimiento, cuyo contenido son artículos científicos y

publicaciones de revistas oficiales del sector de la ingeniería.

Para la revisión sistemática se ha tomado la metodología propuesta por Barbara Kitchenham en 2004 [2]. En esta metodología se considera una revisión sistemática como un método de investigación desarrollado para identificar, obtener, analizar, evaluar, e interpretar toda la investigación existente y relevante para un tema, interrogante, fenómeno o área de interés particular. Un aspecto importante que tiene que cumplir esta revisión es que debe realizarse de forma completa, apoyándose en estudios individuales previos que se van a denominar *estudios primarios*, generando luego la revisión o *estudio secundario*.

Este trabajo se organiza en cinco apartados. El segundo apartado, *Contexto de estudio*, establece las tres áreas de estudio de este artículo: la Computación Orientada a Servicios, a través de la Arquitectura Orientada a Servicios (SOA). Luego, el área relativa al modelado y los lenguajes de modelado, que centraremos en UML y en su aplicación a servicios. La última área de estudio es la que abarca específicamente a las pruebas dentro del campo de servicios.

El tercer apartado, *Proceso de la Revisión*, determina los pasos llevados a cabo para la realización de esta revisión. Se presenta la obtención de información; ésta se ha llevado a cabo en dos fases: la primera se centra en la obtención de información guiada por términos básicos y la segunda usa términos de búsqueda ajustados por los resultados obtenidos de la fase anterior. Se indican las fuentes de conocimiento de las cuales se ha extraído la información, la cláusula de búsqueda y criterios de inclusión y exclusión de los estudios primarios.

El cuarto apartado presenta los resultados, esto es, *los estudios secundarios*. Se validará la calidad de búsqueda de los estudios primarios y se presentarán los trabajos más relevantes en cuanto a su continuidad temporal dentro del campo de pruebas en SOA con UML.

## 2. Contexto de Estudio

El objetivo de este apartado es presentar una introducción a las principales áreas de estudio sobre las que gira esta revisión sistemática.

### 2.1. Computación Orientada a Servicios.

Una de las máximas autoridades en el campo, Michael Papazoglou, define SOA como “un estilo meta-arquitectónico basado en servicios débilmente acoplados que proporciona flexibilidad a los procesos de negocio de manera interoperable e independiente de la tecnología” [3]. El objetivo esencial es por tanto la *interoperabilidad*, y es consecuencia directa del *acoplamiento débil*. Ésta es a su vez consecuencia de los principios esenciales de los servicios SOA, esto es: (1) toda función (simple o compuesta) es descrita como un servicio; (2) todo servicio es independiente de los demás servicios, y sólo se comunica a través de su interfaz, y (3) toda interfaz de servicio puede ser invocada con un identificador único, e independientemente de si es local o remoto. Cada uno tiene más consecuencias: de los dos primeros nace el debate sobre la *composición de servicios* (orquestración vs. coreografía; vertical u horizontal); y del tercero las implicaciones sobre su *descubrimiento* (identificación, contratos, SLA, servicios semánticos), así como el dinamismo y las necesidades de adaptación que éste causa.

La definición del término SOA (*Service-Oriented Architecture*) resulta todavía confusa, a pesar de su popularidad; posiblemente porque se ha utilizado con distintos sentidos en distintos contextos. De manera habitual se utiliza al menos con tres significados distintos, aunque mutuamente relacionados. En primer lugar, SOA como paradigma de computación, esto es, como equivalente de SOC (*Service-Oriented Computing*) [4]. En segundo lugar, SOA como estilo arquitectónico, esto es, como el conjunto de características que debe reunir un diseño para poder calificarlo de este modo. Y en tercer lugar, SOA como referido exclusivamente a la tecnología específica de los llamados *web services*, esto es, los basados en protocolos y estándares como SOAP y WSDL, que sin duda hicieron popular al término; ésta recibe más apropiadamente el nombre de WSA (*Web Services Architecture*). En nuestro caso, y de manera general, se usará el término en la segunda acepción, aunque esto no lo disocia completamente de los otros dos.

SOA tiene un papel especialmente relevante en el contexto de la integración de aplicaciones, el clásico problema de la EAI (*Enterprise Application Integration*). En realidad, puede decirse que nació, como tal, con el objetivo de solucionar este problema. Se plantea SOA como la arquitectura de integración definitiva: todas las capacidades y necesidades de un Sistema de Información heterogéneo se describen como servicios, que se integran en una infraestructura común, el denominado bus de servicios o ESB (*Enterprise Service Bus*). De hecho, en este contexto, SOA y ESB son prácticamente sinónimos. Este enfoque tiene sus consecuencias a otros niveles: la concepción de todo el Sistema de Información a nivel de servicios relaciona a éstos de manera directa con el modelado y gestión de procesos de negocio (BPM, en ambos casos: *Business Process Modeling* o *Management*). Esto se desarrolla tanto a nivel técnico (en la estructura de una orquestación en BPEL, o un servicio semántico en OWL-S) como desde un punto de vista organizativo: en este contexto, el tradicional Gobierno de las TIC se transforma, de manera efectiva, en *SOA Governance*.

Si en lugar de usar SOA, con cualquiera de estas connotaciones, usamos de manera genérica el término servicio –sólo en su acepción moderna en Informática–, la perspectiva es aún más complicada. En su contexto más general, el que usa la SSME (*Service Science, Management and Engineering*), el concepto de servicio describe todo un nuevo paradigma de concepción de los Sistemas de Información, y no sólo de la Informática. En su enfoque más restrictivo, el término se reduce a la tecnología de los web services originales, que más arriba se han denominado como WSA. Por ejemplo, las populares referencias a la “muerte de SOA” se han de entender en este contexto: de hecho, a pesar de enmarcarse en un debate arquitectónico más amplio (SOA vs. REST).

Igualmente, se debe tener en cuenta el papel de los servicios y su composición, a nivel de usuario avanzado, tanto desde la perspectiva de la definición de  *mashups*  personalizados, como de la elaboración de las RIA (*Rich Internet Applications*), tanto en el contexto de la web como en el de las plataformas móviles (teléfonos, PDAs, etc.)

En resumen, los términos a considerar en una revisión sistemática distan de ser simplemente “SOA” o “servicio”, ya que hay que considerar toda una multitud de matices; pero entendidos en su contexto, y teniendo en cuenta su interrelación, se adquiere una perspectiva mucho más amplia de todos los aspectos implicados en este estudio.

## 2.2. Sobre Modelado y el Lenguaje de Modelado Unificado (UML)

UML, concebido como un lenguaje unificado, se ha utilizado también como lenguaje “universal” de modelado, gracias a sus mecanismos de extensión pesada y ligera. Con estos últimos ha tomado la forma de *profiles* UML concretos, que le permiten definir Lenguajes de Modelado Específicos de Dominio (DSMLs).

Una alternativa a los *profiles* de UML es la de crear lenguajes a partir del metamodelo de alto nivel definido por el MOF, como es el caso del *Eclipse Modeling Framework* (EMF) de Eclipse. Esta herramienta ha sido la elegida en muchos desarrollos de software, lo que ha hecho que muchas aplicaciones no trabajen realmente con UML, aunque sí haya un grado de compatibilidad debido a que UML también utiliza el MOF.

En la literatura la utilización de UML en el proceso de pruebas es indiscutible. La utilización más frecuente es la generación de casos de prueba a través de los diagramas de secuencia y los diagramas de estado. Pero en muchos casos, la utilización de algunos diagramas de UML en el tema de pruebas va aún más allá: permiten modelar todo el proceso de pruebas, haciendo uso de un *profile* de pruebas definido desde UML. De hecho, el *profile* UTP [5] se está consolidado como un estándar de la OMG.

La utilización del modelado en el proceso de desarrollo de software permite la generación de forma automática de aplicaciones mediante las transformaciones de modelos, descritas utilizando lenguajes tales como ATL o QVT; éste es el enfoque definido por MDA (*Model-Driven Architecture*). Cabe preguntarse la existencia de alguna técnica o aproximación que utilice los principios de MDA en el campo de pruebas. En efecto así es; durante la búsqueda de

resultados sobre pruebas y SOA, se encuentran dos líneas relacionadas con el tema de modelado: Las pruebas basadas en modelos, *model-based testing*, y dentro de ésta, ligada específicamente a MDA, estaría el *model-driven testing*. Estas dos técnicas de pruebas también se han localizado en el proceso de búsqueda de esta revisión sistemática.

El tema de modelado aplicado al área de los servicios esta representando a través de una metodología propia de SOA, IBM SOMA (*Service-Oriented Modeling and Architecture*). Esta metodología utiliza una técnica de análisis y diseño denominada SOAD [6], adaptada de la orientación a objetos y ubicada dentro del marco de modelado SOMF (*Service-Oriented Modeling Framework*). La utilización de SOAD no rompe con el lenguaje de modelado unificado, a pesar de tener un lenguaje característico propio. Esto es, a nivel de aplicación, el uso de UML se mantiene; es en el nivel de arquitectura del sistema en el que se aplican consideraciones particulares propias de la orientación a servicios, utilizando el lenguaje de modelado SoaML, que está siendo a su vez considerado como estándar de la OMG. A su vez, SoaML no supone una ruptura con UML, sino una extensión a través del mecanismo de *profiles*.

### 2.3. Pruebas sobre Sistemas Orientados a Servicios

Las propiedades intrínsecas a la orientación a servicios hacen que el tratamiento de pruebas tenga matices diferentes a la orientación a objetos. Así, la dificultad en conocer el código y su estructura complica el poder establecer pruebas estructurales basadas en la cobertura de ejecución. Por ello resulta más compleja la realización de pruebas de caja blanca, si bien hoy existen propuestas interesantes como la de Bartolini [7].

Además de la dificultad anterior, si se consideran las pruebas no sólo a través del código, hay más aspectos que hacen difícil probar los sistemas orientados a servicios, como indica Canfora [8]. Su dinamismo y la adaptabilidad intrínseca complican las pruebas. El control de los servicios resulta complicado, debido a que como tal, no se integran en el

sistema, sino que se ejecutan en una infraestructura independiente que controla el proveedor del servicio. La veracidad de las características de calidad y el coste que suponen las pruebas son otros de los aspectos que complican todo este proceso.

Algunas referencias de interés que cubren el tema de pruebas en el campo de servicios son las que resumen las propuestas de Canfora [8] y Bucchiarone [9], este último a nivel de composición de servicios.

Como se ha mencionado con anterioridad, dentro del campo se encuentran también los sistemas de pruebas basados en modelos, *model-based testing* (MBT) [10]. Tal como se indica a continuación, estas referencias fueron el punto de partida de la siguiente fase de búsqueda.

Los niveles a los que se aplican las pruebas en un contexto de servicios son principalmente a nivel de unidad, es decir a nivel del servicio, y a nivel de integración o composición de servicios, esto es, pruebas de integración.

## 3. Proceso de Revisión

### 3.1. Motivo y Fuentes de Conocimiento

El motivo que ha llevado a plantearse esta revisión es comprobar las posibilidades que existen, o pudieran existir, de utilizar modelos UML como base para diseñar pruebas de software en el ámbito de la arquitectura software orientada a servicios. Este interés está apoyado en una de las primeras propuestas para el ciclo de desarrollo de sistemas SOA donde se integraban modelos UML como soporte de análisis y diseño [11].

Para ello, la principal base de información sobre la que se ha trabajado, se ha obtenido de la Universidad de Alcalá de Henares (UAH), mediante el acceso a las Bases de Datos de artículos científicos. Ésta, a día de hoy, dispone de un total de más de 400.000 volúmenes, unos 5.300 títulos de publicaciones seriadas en diferente tipo de soportes y recursos electrónicos: acceso a bases de datos y un total de 13.000 títulos de revistas electrónicas. Es este último recurso el que se ha tomado como punto de partida para este estudio.

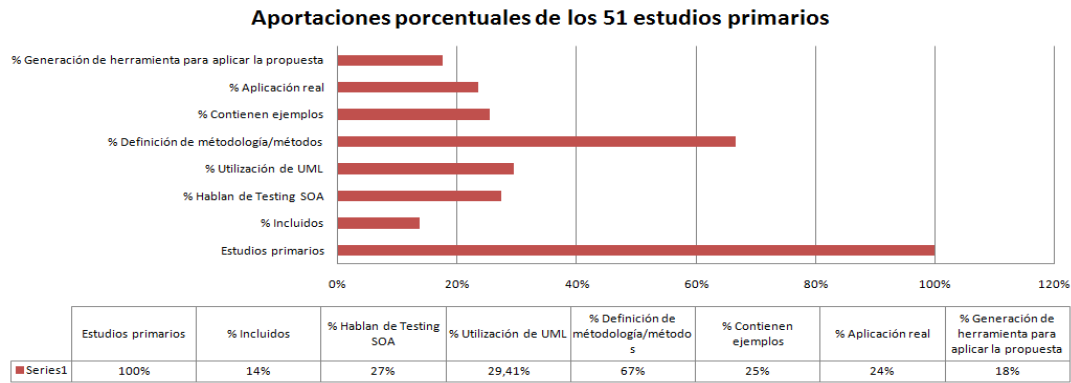


Figura 1. Aportaciones porcentuales de estudios primarios obtenidos en la primera fase

La búsqueda de la información se facilitó gracias a un Buscador, gestor de enlaces SFX, incorporado en el portal “MetAL” -que integra todos los recursos electrónicos de la UHA-, junto con los servicios suministrados por los editores y proveedores de las suscripciones.

La búsqueda de información se guió por las siguientes preguntas:

5. ¿Existen metodologías, procesos y herramientas definidas para el aseguramiento de la calidad mediante el estudio de las pruebas de software orientado a servicios?
6. ¿Cuáles son los métodos más conocidos que siguen los estándares de UML?
7. ¿Hay casos de estudio o casos de prueba realizados en estos sistemas?

El proceso de obtención de información se realizó en dos fases: En una primera fase se optó por buscar información con términos básicos implicados con la búsqueda. En una segunda fase, con información refinada obtenida de la primera fase, se optó por localizar información más específica, para mejorar las conclusiones y hacer un mejor *estudio secundario*. Así, la principal consulta para obtener los estudios primarios estuvo formada por las siguientes cláusulas lógicas, enlazadas por el operador OR:

- [(“Test” OR “Testing”) AND (“SOA” OR “Service oriented architecture” OR “Web service” OR “UML” OR “unified modeling language” OR “web” OR “methodology”)].

- [(“SOA” OR “Service oriented architecture”) AND (“UML” OR “unified modeling language”)].
- [(“Web service”) AND (“UML” OR “unified modeling language”)].

En un primer momento con estas cláusulas pretendíamos conseguir la mayor información posible dentro del campo. A posteriori, y dado que, en otras áreas lejanas al campo de la ingeniería software se hace uso de este tipo de términos, se optó por restringir el campo de estudio al ámbito de tecnología de la información y considerar sólo los estudios posteriores al 1995, fecha en la que aparece la primera versión de UML.

En la segunda fase la consulta se realizó con las cláusulas siguientes, tras el análisis de los resultados obtenidos en la primera Fase:

- (“Model based testing” AND “Service Oriented”) y sus variantes.
- (“Model driven testing” AND “Service Oriented”) y sus variantes.

En esta segunda fase los resultados se focalizaron en búsquedas de los editores que se presentan a continuación, ya que estos habían sido los que habían producido resultados en la fase anterior: ACM (Association for Computing Machinery), IEL/IEEE (Institute of Electrical and Electronics Engineers), ScienceDirect (Elsevier), SpringerLink (Kluwer).

### 3.2. Criterios de Inclusión y Exclusión

La revisión de los estudios primarios se hizo de forma exhaustiva haciendo en un primer momento una lectura rápida, que permitió decidir el grado de profundidad de la lectura según se ajustara o no a los criterios de inclusión o exclusión. Independientemente de esto, para la realización de la revisión se procedió a la realización de una ficha de cada uno de los estudios primarios obtenidos.

Los criterios de inclusión se centraron en artículos que trataban aspectos de “test” o aseguramiento de la calidad para aplicaciones orientadas a servicios, haciendo mención directa de aquellos artículos que utilizan UML en el ámbito de pruebas. La lista de tópicos relevantes para la inclusión se recoge en la siguiente lista:

- Test para SOA.
- Métodos para el proceso de “testing” en SOA.
- Estándares para el proceso de “testing” en SOA.
- UML en la fase de pruebas.
- Referencia y ejemplifica modelos UML para SOA y su desarrollo.
- Propone herramientas para el uso de UML en pruebas para SOA.

#### 4. Resultados y Análisis de la Revisión.

En este apartado se presentan los resultados obtenidos de la primera y segunda Fase, finalizando con un resumen del *estudio secundario* obtenido.

##### 4.1. Fase 1: Resultados y Análisis

Los resultados de la revisión se presentan en la Figura 1. Los campos que se tienen en cuenta tanto para esta figura como los de la tabla 1 son:

- **Nº:** Hace referencia a la clave de revisión de estudio primario.
- **Título:** Es el título completo del artículo.
- **Inclusión:** Hace referencia a si cumple los requisitos de inclusión
- **SOA Testing:** Indica si se utilizan pruebas para aplicaciones SOA.
- **UML:** Si utilizan el lenguaje de modelado unificado.
- **Metodología:** Propone una metodología o un método de actuación.

- **Aplicación práctica:** Documenta ejemplos o casos de aplicación
- **Herramientas:** Hace uso o construye alguna herramienta para el método

Se excluyeron del estudio detallado, en la primera fase, aquellos artículos que sólo hacían referencia a procesos, pero no dan información adicional de cómo se lleva a cabo las pruebas. No se excluyeron los artículos que, tratando el tema de pruebas en orientación a servicios, no utilizaban UML. El objeto de no haberlos descartado en esta primera fase fue poder comprobar si UML se había descartado por la utilización de otros lenguajes o otros métodos. En la segunda fase se excluyeron de la búsqueda aquellos que no estaban directa o indirectamente ligados con UML

Así para el 100% de los estudios primarios obtenidos sólo se consideraron incluidos el 14%. El 27% habla de “testing” con SOA, el 29,41% utiliza UML, el 67% hace una metodología o aplica un método, el 25% contiene ejemplos, el 24 % aplica los resultados sobre un entorno real. Para el estudio se utilizaron un total de 21 editores de los 44 disponibles y de las 1577 revistas publicadas por dichos 44 editores, al final fueron útiles un total de 1545. Así, un total de un 47,72 % de los editores fue útil para nuestro estudio y del total de revistas que nos proporcionaron dichos editores, se utilizaron para el estudio un total del 97,97%.

De los 1545 artículos útiles y descartando los que trataba los términos SOA, test, UML de forma lateral, se obtuvieron un total de 51 artículos. De estos, y tras un análisis para considerar aquellos que incluyeran métodos, o metodologías estándares o aplicaciones prácticas y herramientas, se obtuvieron un total de 7 estudios relevantes. Sólo dos [12], [13] incluían todos los aspectos propuestos, sin considerar la utilización de herramientas.

##### 4.2. Fase 2 : Resultados y Análisis

Una característica común a ambos artículos es que se centran en las técnicas de pruebas basadas en modelos, *model-based testing*, al que se ha hecho referencia en el apartado 2.3. En el artículo [12] la primera parte está dedicada a la definición de la propuesta a grandes rasgos y la

Título	Autores	Descripción
Hybrid test of web applications with webtest (2008)	Harald Raffelt et al	-Sistemas de pruebas estático y Sistema de pruebas dinámico con técnicas de aprendizaje basadas en autómatas. -Considera pruebas de caja negra. Futuro considera posibilidad de incorporar TTCN-3 a la herramienta WebTest, - Aplicación en RIAS sin servicios
Engineering rich internet applications with a model-driven approach (2010)	Piero Fraternali et al	Webtest herramienta utiliza dos framework: -Jabc cubre aproximacion model-drive y orientada a servicios - Proceso de prueba dinámico, facilidad de almacenar, reproducir, integrar y recoger información.
Towards Adaptive Test Code Generation for Service Oriented Systems (2009)	Felderer, M., et al	- <i>Profile</i> propio orientado al negocio concreto compatible con UTP por transformación sin pasar por TTCN-3 y adaptable a SOAML (base de aplicación en UML). Comprobaciones de consistencia/recubrimiento, generación, ejecución y análisis pruebas. Se ejecutan automáticamente por un <i>framework</i> caso concreto FIT. Transformación de modelos y Aserciones
Testing Service-Oriented Architecture Applications (2007)	Paul Baker et al	- Profile UML para Pruebas UTP. -Frameworks TTCN-3 y el JUnit test framework para Java. -Demuestra como usar UTP con el framework TTCN-3 y el framework de prueba JUnit para Java
A Model-Driven Approach to Discovery, Testing and Monitoring of Web Services (2007)	Marc Lohmann, Leonardo Mariani y Reiko Heckel	UML+Transformaciones de grafos y Restricciones OCL para Java modeladas con JML. 1. Extiende y aplica <i>domain-based testing</i> y técnica de flujo de datos para transformación de grafos. 2. Genera pruebas ejecutables predecibles desde la transformación de grafos. 3. Prueba y valida automáticamente servicios web, antes de incorporarlos al sistema
Model-based functional conformance testing of web services operating on persistent data (2006)	Avik Sinha, Amit Paradkar	Propone un sistema de <i>model based test case</i> para la generación de casos de prueba que permite generate funcional conformance casos de prueba para servicios web basado en Extended Finite State Machine (EFSM). Primero se especifican con el estándar WSDL-S, y luego se traslada a EFSM.

Tabla 1. Selección de artículos de la fase dos, complementarios a los MBT y MDT del texto

definición de todos los conceptos que rodean a la generación del Framework de testeo.

En primer lugar se hace una introducción al lenguaje *Testing and Test Control Notation* (TTCN-3). El Framework propuesto consta de una serie de casos de test para la evaluación de la carga y funcionamiento de servicios Web.

En [13] la aplicación de la propuesta, en torno a la que gira todo el artículo, es el uso de las técnicas y el framework propuesto sobre la aplicación *The eHealth Alarm Dispatcher*, encargada de comunicar el dispositivo móvil de una persona con los servicios médicos mediante llamadas a Web Services.

El artículo describe y define el *framework* de testeo que proponen desde este proyecto, así como el modelado de sistemas a través de la representación en diagramas denominados *Symbolic Transition System* (STS).

La información para generar estos diagramas se obtiene de la definición de los servicios Web, normalmente como un fichero WSDL; para ello, se ha desarrollado la herramienta MINERVA.

Como punto final se encuentra la descripción de una herramienta desarrollada en Java, *Jambition*, que va a generar los casos de test a partir de los diagramas STS. Todo ello se realiza dentro del Proyecto PLASTIC.

Los resultados de esta segunda fase están ligados a las dos técnicas de pruebas: *model-based*, y *model-driven testing* (MBT, MDT).

En esta segunda fase, que se reorienta la búsqueda hacia las dos técnicas anteriores con servicios y UML, se obtuvieron un total de 55 artículos. En este caso la búsqueda se centra en los 4 editores más representativos encontrados en la primera fase. Se comprueba que todos ellos están en un margen temporal entre 2006 y el 2010. En esta búsqueda se obtuvieron un total de 9 artículos directamente implicados con nuestro estudio, de los 55 obtenidos en la búsqueda de esta segunda fase. El mayor número de artículos se concentra en torno al 22% entre el año 2008 al 2010. Dentro de los artículos se puede considerar una línea de trabajo continuada en el grupo de artículos firmados por Antonia Bertolino y los miembros de los proyectos

TAROT y PLASTIC. Este grupo trabaja en la línea de pruebas a nivel de integración y de sistema con descripciones arquitectónicas y modelos UML. Dentro de este grupo una referencia interesante es la de SOA Governance [15]. Lo más interesante es su metodología de trabajo sustentada por herramientas.

Otra línea de interés, dentro del *model-driven testing*, es en la que trabaja el grupo de Ina Schieferdecker [16]. Las líneas en las que trabajan es en la especificación utilizando UML, en *Message Sequence Charts* (MSC) y en *Testing and Test Control Notation* (TTCN-3). Otras aproximaciones de MBT y MDT se presentan en la Tabla 1<sup>1</sup>.

## 5. Conclusiones

El campo de pruebas en la orientación a servicios es un área aún por explorar. Las características peculiares de estos sistemas no permiten la aplicación directa de técnicas tradicionales, sin más cambios. Se requiere hacer adaptaciones en el campo de pruebas a través de modelos, donde se aprovechan las capacidades de los diferentes lenguajes y sus ampliaciones como es el caso de UML a través de los *profiles* o los metamodelos. Se han de considerar los lenguajes implicados a diferentes niveles; para el caso de modelado de aplicación podríamos aplicar UML y adaptaciones, pero para el nivel de arquitectura del sistema se requeriría el uso de lenguajes específicos, donde el más significativo es BPEL.

## Agradecimientos

Los autores agradecen el apoyo del proyecto TIN2007-67843-C06-01 para este trabajo.

## Referencias

- [1] Kitchenmam, B., *Procedures for performing systematic review*. 2004, Department of Computer Science, Keele University.
- [2] Papazoglou, M. P. *Web Services: Principles and Technology*. Pearson, 2008

- [3] Papazoglou, M. P. *Web Services: Principles and Technology*. Pearson Pearson, 2008.
- [4] Bell, M. *Service-Oriented Modeling: Service Analysis, Design, and Architecture*. John Wiley & Sons, 2008.
- [5] Object Management Group. RFP for a UML Testing Profile <http://www.omg.org/>
- [6] Papazoglou, M. P.; Georgakopoulos, G. *Service-Oriented Computing*, Communications of the ACM, 46(10), pp.24-28, Oct. 2003.
- [7] Bartolini, C.; Bertolino, ; Elbaum, S and Marchetti, E. Whitening SOA testing. In V. Issarny, editor, ESEC-FSE '09. ACM, 2009.
- [8] Canfora, G and Di Penta, M. "Service-Oriented architectures testing: a survey," LNCS, 2009, vol. 5413
- [9] Bucchiarone, A.; Melgratti, H.; Severoni, F. Testing service composition. In: (ASSE'07), Mar del Plata, Argentina, 2007.
- [10] Dias Neto, A.C; Subramanyan, R; Vieira, M and Travassos, GH. A survey on model-based testing approaches: a systematic review. In WEASEL'07, pages 31, 36, Atlanta, Georgia, 2007. ACM
- [11] Erl, T.: *Service-Oriented Architecture: Concepts, Technology & De*, Pearson, 2005.
- [12] Schieferdecker, I et al. Distributed Functional and Load tests for Web services, (STTT), Springer 2004
- [13] Frantzen, L. et al. On-The-Fly Model-Based Testing of Web Services with Jambition, Workshop, WS-FM 2008, Milan, Italy, September 4-5, 2008, Springer-Verlag, Berlin, Heidelberg, 2009
- [14] Bartolini, C.; Bertolino, A.; Elbaum, S. and Marchetti, E. Whitening SOA testing. In V. Issarny, editor, ESEC-FSE '09. ACM, 2009.
- [15] Bertolino, A and Polini, A. SOA test governance: enabling service integration testing across organization and tech. borders. In *Wor. on Web Testing*, pp. 277-286, 2009.
- [16] Zander, J., Dai, Z.R., Schieferdecker, I., Din, G.: From U2TP models to executable tests with TTCN-3: an approach to model driven testing. In: *Testing of Communicating Systems*. LNCS, vol. 3502/2005, page. 289–303. Springer (2008).

<sup>1</sup> Las referencias de la tabla no están listadas en el apartado de referencias por problemas de espacio.