Análisis y Verificación de un Sistema de servicios Web para la mejora de regadíos *

Juan José Pardo, Gregorio Diaz, M.Carmen Ruiz, M.Emilia Cambronero, and Valentín Valero

Departamento de Sistemas Informáticos
Universidad de Castilla-La Mancha
Escuela Politécnica Superior de Albacete. 02071 - SPAIN
[jpardo,gregorio,carmenr,emicp,valentin]@dsi.uclm.es

Resumen En este articulo presentamos un trabajo que actualmente se encuentra en curso y en el que se realiza una modelización y análisis de un sistema de control de riegos basado en servicios Web. Para conseguir nuestro objetivo el sistema es modelado mediante autómatas temporizados que serán verificados mediante la herramienta UPPAAL que nos permitirá validar y verificar el sistema.

1. Introducción

Desde la aparición de Internet el mundo de la informática ha sufrido un cambio radical. La primera revolución acontecida ha sido el desarrollo de Internet como una gran fuente de información. Dónde se ha producido un gran intercambio intercultural y socioeconómico que ha propiciado lo que hoy conocemos como globalización. Aunque, como de costumbre en todos los oficios podemos hablar que su propio arte no suele redundar e beneficio de facilitar las labores que conllevan a su consecución. De ahí la frase célebre "En casa del herrero cuchara de palo". Algo parecido ha ocurrido en nuestro caso con Internet, Una gran cantidad de gente se ha aprovechado de su uso pero los ingenieros informáticos, no hemos sido capaces de usar esta potente herramienta más allá de usarla como una biblioteca o como mucho una plataforma de intercambio de datos.

Es así que en los últimos años se están desarrollando nuevas tecnologías basadas en la facultad que tiene Internet para el intercambio de datos. La idea que subyace es utilizar esta facultad para poder crear aplicaciones complejas a partir de otras aplicaciones más sencillas. Una de estas nuevas tecnologías es conocida como Servicios Web. La definición de un Servicio Web puede ser la de una unidad o bloque de código que puede interactuar con otro u otros servicios Web creando de esta forma una aplicación.

Esta nueva tecnología actúa por encima de las capas http, ftp, smtp, etc. que ofrecen Internet y está basada en tres protocolos básicos SOAP, WSDL y UDDI. El protocolo SOAP establece un entorno XML en el cual viajan los datos que son intercambiados

* Este trabajo ha sido parcialmente financiado por el proyecto CICYT "Web Services y Testing: Fundamentos y Aplicaciones (WEST)" TIN2006-15578-c02-02 y el proyecto de la JCCM "Aplicación de métodos formales al diseño y análisis de servicios Web y comercio electrónico"PAC06-0008-6995.

por los servicios Web. El protocolo WSDL es el lenguaje en el cual es codificada la información acerca de como se debe establecer la comunicación entre los diferentes servicios. Y por último, el protocolo UDDI nos permite descubrir los servicios que ofrece un servidor. En la figura 1 podemos observar el esquema de comunicación que se establece entre los diferentes actores de este sistema de intercambio de información.

Figura 1. Diagrama de funcionamiento de los servicios Web.

Así pues esta tecnología se está extendiendo debido a las ventajas que ofrece este entorno distribuido. Un ejemplo claro de estas ventajas es el ahorro que supone el uso compartido de aplicaciones, si entendemos una aplicación como un servicio Web que pueda ser usado por otros servicios-aplicaciones. El ejemplo típico podría ser una base de datos meteorológica que nos ofrezca mediante un servicio Web la información del tiempo en diferentes lugares. Nosotros podríamos ofrecer en nuestra aplicación dicha información y no sería necesario mantener toda una base de datos para acceder a este tipo de información.

Pero esta tecnología posee a su vez de una serie de inconvenientes, y uno de ellos es el aumento de la complejidad en la comunicación de dichos sistemas. Un campo que lleva décadas estudiando dicha complejidad es el campo de los métodos formales. Dicho campo permite realizar estudios sobre la interacción de dichos procesos-servicios que actúan de forma independiente y paralela. Con ellos podemos comprobar que el diseño de dichos sistemas cumple una determinada serie de requisitos. Estos requisitos pueden ser tanto propiedades funcionales como no funcionales que el sistema debe acometer para su apropiado funcionamiento.

Existen diversas técnicas que nos permiten realizar esta tarea pero entre ellas podemos destacar unas que nos ofrece herramientas ampliamente extendidas con las que trabajar, estamos hablando del Model Checking. El Model Checking permite modelar los sistemas a verificar, especificar los requisitos mediante propiedades y comprobar si el modelo del sistema cumple las propiedades o en caso contrario obtener un contrajemplo para poder localizar el error. Para esta técnica se han desarrollado numerosas herramientas como SPIN, KHRONOS, UPPAAL y muchas otras. Nosotros, en nuestro caso, nos centraremos en la herramienta UPPAAL pues los sistemas que deseamos estudiar poseen restricciones temporales. Esta herramienta posee relojes y otras variables que nos permitirán modelar los comportamientos temporales del sistema y además la lógica que utiliza para la especificación de propiedades permite restricciones sobre los relojes.

Así pues, como ya hemos comentado, el Model Checking o comprobación de modelos se basa en varias fases que son: **modelar** el sistema, **especificar** las propiedades-requisitos del sistema, **comprobar** dichas propiedades sobre el modelo y si alguna propiedad no se cumple obtener un **contraejemplo** que mediante un **simulador** localizará donde se ha producido el error. UPPAAL utiliza autómatas temporizados para modelar el sistema, una lógica temporal basada en TCTL para especificar las propiedades, un algoritmo de comprobación de las propiedades basado en la búsqueda de estados simbólicos que representan el valor de los relojes y variables mediante restricciones ($=$, $<$, $>$, $\leq$ y $\geq$) y por último un simulador que permite cargar las trazas generadas por los contraejemplos para localizar dónde podría estar el error. Para ahondar con más detalle en los entresijos de esta herramienta el lector puede consultar la numerosa documentación disponible en la URL: http://www.uppaal.com.

Una vez bosquejado el principal objetivo de este trabajo, podemos pasar a definirlo con mayor precisión. En este trabajo pretendemos aplicar el Model Checking con la herramienta UPPAAL para realizar la evaluación de un sistema diseñado mediante servicios web. El sistema bajo estudio consiste en una aplicación software con esquema cliente-servidor dónde la funcionalidad del servidor se ha implementado mediante servicios web. y más concisamente, el servidor es un servicio web de información geográfica (GIS) que un cliente web puede consultar para obtener información de publiometría, humedad relativa y otros datos necesarios para gestionar zonas de riego de una manera más eficiente, con el consiguiente ahorro en recursos hídricos. Este sistema ha sido ideado y se está llevando a cabo en el Instituto de Desarrollo Regional de la Universidad de Castilla-La Mancha y tiene como objetivo brindar a regantes y administraciones una potente herramienta que facilita la explotación de recursos hídricos de una forma eficiente y correctamente regulada.

Este trabajo está dividido en 4 apartados. En la sección 2 se realiza una descripción de las principales características del sistema objeto de estudio para en la sección 3 realizar el modelo del mismo. A continuación en el apartado 4 se presentan algunas de las propiedades que se han verificado para el sistema y finalmente se presentan unas conclusiones y las lineas por las que va a continuar este trabajo.

2. Descripción del sistema

El sistema diseñado tiene una estructura cliente/servidor con un motor de servicios web que ofrece toda la funcionalidad del sistema y que ejerce de interfaz entre el usuario cliente y el servidor. Así este motor recibe las peticiones de servicios realizadas por los diferentes clientes, y tras obtener la información necesaria, hace la petición del servicio al servidor. Una vez solicitado el servicio la tarea de la pasarela habrá terminado y los datos obtenidos serán enviados directamente desde el servidor al cliente para que este los muestre. La comunicación entre los diferentes agentes que intervienen en el sistema (clientes, pasarela y servidor) se realiza mediante XML.

Los servicios proporcionados por el sistema son los siguientes.

sLogin . Es utilizado por el cliente para solicitar un identificador de sesión que le permita utilizar el resto de los servicios del sistema. Para la obtención de este identificador de sesión el usuario debe proporcionar su usuario y clave del sistema. El identificador de sesión tiene una validez de 30 minutos que una vez transcurridos sin actividad hacen necesario la solicitud de un nuevo identificador de sesión.

sGetFarmsUSer .Proporciona información de las fincas a las que tiene acceso el usuario que realiza la petición. Para obtener esta información el servicio debe interactuar con el gestor de la base de datos.

sGetPlotsFarms . Proporciona información acerca de los plots asociados con una determinada finca indicada por el usuario.

sGetStations .Proporciona un listado de las estaciones meteorológicas que corresponden al usuario que utiliza el sistema. La información obtenida se utiliza para completar la información proporcionada por el cliente de la aplicación al usuario.

sGetMap .Es uno de los servicios que más funcionalidades presta al usuario siendo la más básica de ellas la de generación de mapas para ser mostrados. Tras la verificación de los permisos y con la información suministrada por el cliente sobre el mapa deseado este servicio prepara un fichero de configuración que pasa al servidor de mapas para que éste genere el mapa. Tras esto el servicio termina su tarea y será el servidor de mapas el que envíe directamente la información sobre el mapa al cliente.

sSelectFeature . Permite obtener información sobre las parcelas contenidas en una ventana definida sobre la imagen que muestra el cliente.

sGetPixelValue . Permite obtener el valor de un pixel de la imagen para una fecha solicitada.

sGetPixelTSTable y sGetPixelTSGraph . El sistema proporciona estos dos servicios para la obtención de los valores de un determinado pixel de la imagen durante un periodo temporal. La diferencia entre ellos radica en la forma de presentación de los datos que el primero de ellos lo hace en formato tabular mientras que el segundo lo realiza mediante un gráfico.

sGetTimeSerieTable y sGetTimeSerieGraph . Proporciona los valores de un parámetro numérico (media, desviación estándar, etc.) Al igual que ocurría antes el primero de los servicios lo muestra en formato tabular mientras que el segundo lo muestra en formato de gráfico.

sSelectByQuery Al igual que el servicio sSelectFeature este servicio nos permite obtener la información sobre un conjunto de parcelas cuya elección en este caso se realiza mediante las características que deben cumplir y no mediante un recuadro como ocurría antes.

sGetFeatureInfo . Proporciona una de las funcionalidades más básicas de un SIG como es obtener toda la información relevante para una parcela en una fecha indicada.

sGetASCrops . Devuelve al usuario el centro de Asesoramiento de Riegos que le corresponde y los cultivos sobre los que ese centro puede asesorar.

sGetReportTable y sGetReport . Son dos servicios complementarios donde el primero proporciona toda información que necesita el cliente para genera un informe sobre un conjunto de parcelas. Este informe esta compuesto de tres elementos, una tabla, un gráfico y diversos mapas que contendrán toda la información. El segundo se encarga de, a partir de los datos proporcionados por el primer servicio, de generar un fichero pdf con el informe.

3. Modelado del sistema

De la descripción del sistema podemos obtener que en el mismo intervienen 5 componentes,

- El cliente web que se encarga de realizar la solicitud de los servicios y de proporcionar los datos necesarios para que el sistema pueda proporcionar la información solicitada. El autómata que modela el comportamiento de este elemento es el mostrado en la figura 2

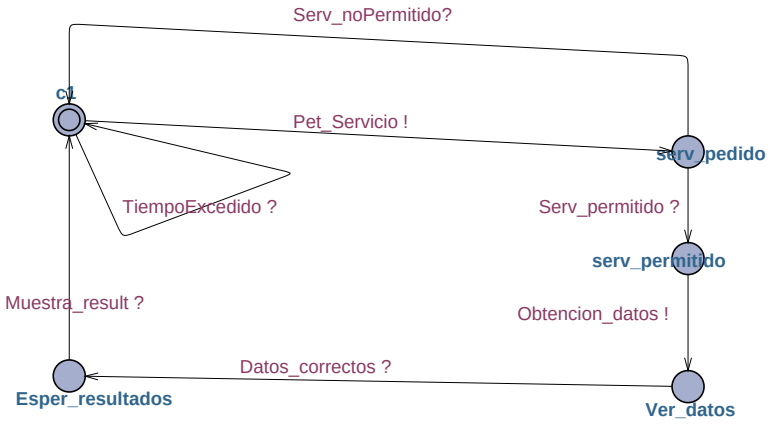

Figura 2. Autómata para el cliente web.

- La pasarela es la encargada de las comprobaciones iniciales ante cualquier petición y de realizar la llamada al servidor para que realice la tarea indicada. Su comportamiento queda modelado en el autómata mostrado en la figura 3

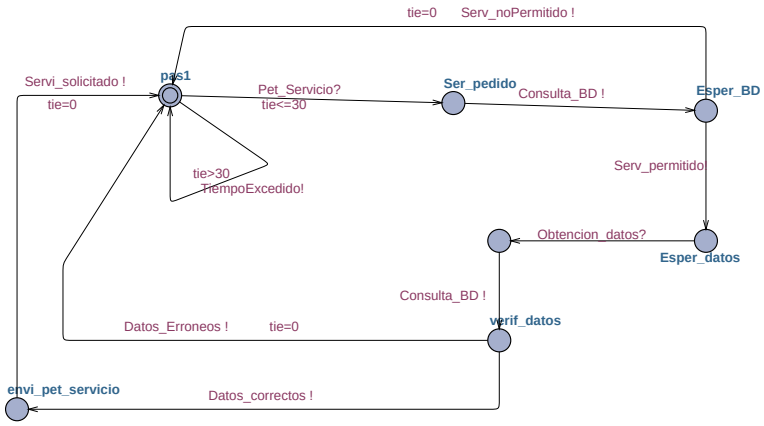

Figura 3. Autómata de la pasarela

Como puede observarse en esta especificación ha sido necesario la introducción de un reloj que nos permita modelar el periodo de validez de la sesión. Así, este reloj es reseteado a cero después de la conclusión del proceso asociado a una solicitud de servicio por parte del cliente.

- El servidor que se encarga de ejecutar las acciones del servicio solicitado. Si se realiza un estudio pormenorizado de cada uno de los servicios que ofrece el sistema podemos comprobar que el comportamiento del servidor en todos los casos es muy similar variando solamente la información intercambiada. Por este motivo en lugar de realizar un modelado de cada uno de los servicios se ha realizado un modelo genérico aplicable a todos los servicios y así se ha obtenido el autómata mostrado en la figura 4
- Tanto la base de datos como el servidor de mapas son dos componentes cuyo comportamiento es muy simple, se limitan a proporcionar la información solicitada por el servidor. Los autómatas obtenidos para cada uno de estos componentes son los mostrados en las figuras 5 y6 respectivamente.

4. Análisis y Verificación

Para el análisis y verificación hemos utilizado la herramienta UPPAAL. Una vez realizado el modelo del sistema procedemos a su verificación. Para ello en primer lugar realizamos una simulación del mismo. Ésta nos permite comprobar que el comportamiento del sistema es el deseado y que el flujo de información entre los componentes es el adecuado.

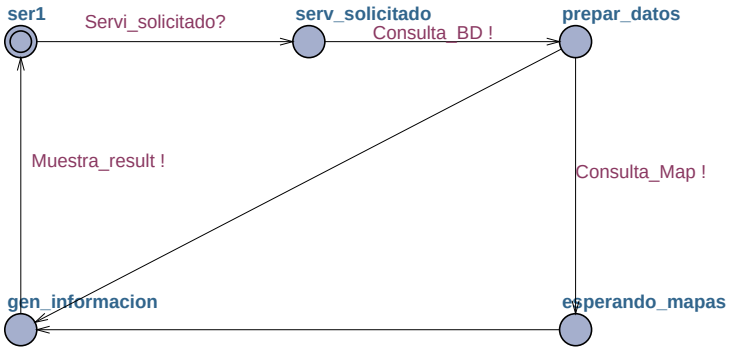

Figura 4. Autómata del servidor.

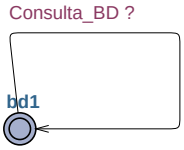

Figura 5. Diagrama de funcionamiento de los servicios web.

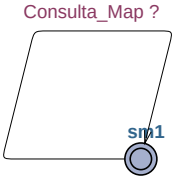

Figura 6. Diagrama de funcionamiento de los servicios web.

Tras la simulación es interesante comprobar que el sistema cumple diversas propiedades. Estas las podemos clasificar en propiedades de vivacidad, seguridad y ausencia de bloqueos.

Ausencia de bloqueos . Se puede comprobar que nuestro sistema esta libre de bloqueos.

Propiedades de Vivacidad .Dentro de este grupo incluimos aquellas propiedades que nos demuestren que nuestro sistema se comporta de una forma lógica. Para nuestro sistema las propiedades de vivacidad son muy simples. Entre ellas podemos destacar la siguiente.

- Si un cliente realiza una petición la pasarela realiza una consulta a la base de datos. Para ello utilizamos la siguiente fórmula.

$$cliente.realiza\_peticion \longrightarrow pasarela.consulta\_Basedatos$$

Propiedades de seguridad . Entre las propiedades de seguridad que debe cumplir nuestro sistema se encuentra la de que una petición de servicio es aceptada solo si se realiza antes de 30 minutos de la anterior. Esta propiedad puede ser modelada mediante la siguiente formula de la lógica temporal

$$\forall \Box cliente.realiza\_peticion \wedge tie <= 30$$

5. Conclusiones

Con el presente trabajo se ha querido aplicar la metodología de model-checking para la verificación de un sistema de control de riegos basado en un sistema GIS. Debido al estado inicial de este trabajo se abre una gran perspectiva de trabajo futuros entre las que podemos destacar la realización de un estudio exhaustivo y detallado del sistema que nos permita detectar nuevas restricciones temporales en el mismo así como refinar el modelo del sistema. Igualmente se realizará una modelización mediante redes de Petri que nos permita realizar el análisis de otras propiedades que con los autómatas resulta más complejo.

Referencias

1. R. Alur and D. Dill, *Automata for modeling real–time systems*, In Proceedings of the 17th International Colloquium on Automata, Languages and Programming, volume 443, Editors. Springer–Verlag, 1990.
2. Assaf Arkin, Sid Askary, Ben Bloch, et. al., *Web Services Business Process Execution Language Version 2.0*, Editors. OASIS Open, December 2004. In http://www.oasis-open.org/committees/download.php/10347/wsbpel-specification-draft-120204.htm
3. Mario Bravetti, Claudio Guidi, Roberto Lucchi and Gianluigi Zavattaro , *Supporting E-commerce system formalization with Choreography Languages*, In Proc. of the 20th ACM Symposium on Applied Computing (SAC'05), special track on E-Commerce Technologies , ACM Press, 2005.
4. Edmund M. Clarke and Jr. and Orna Grumberg and Doron A. Peled, *Model Checking*, MIT Press, 1999.

5. G. Diaz, J. J. Pardo, M. E. Cambronero, V. Valero and F. Cuartero, *Verification of Web Services with Timed Autoamata*, In proceedings of First International Workshop on Automated Specification and Verification of Web Sites, Valencia, March 2005.
6. G. Diaz, J. J. Pardo, M. E. Cambronero, V. Valero and F. Cuartero, *Automatic Translation of WS-CDL Choreographies to Timed Automata*, In proceedings of WS-FM, 2005.
7. Constance Heitmeyer and Dino Mandrioli. *Formal Methods for Real-Time Computing*. John Wiley & Sons. 1996.
8. Nickolas Kavantzas et al. *Web Service Choreography Description Language (WSCDL) 1.0*. In http://www.w3.org/TR/ws-cdl-10/
9. K. Larsen and P. Pettersson and Wang Yi, UPPAAL *in a Nutshell*, Int. Journal on Software Tools for Technology Transfer, Editors. Springer–Verlag vol.1, 1997.
10. Simon Woodman, et al., *Specification and Verification of Composite Web Services*, In proocedings of The 8th Enterprise Distributed Object Computing Conference 2004.