Diseño e Implementación de un Servicio Web para la Simulación de la Propagación de Incendios Forestales

Pazos Ferreiro, J.A., Varela Pet, J., Ríos Viqueira, J.R., Cotos Yánez, J.M.

Laboratorio de Sistemas, Instituto de Investigaciones Tecnológicas,
Universidad de Santiago de Compostela
Constantino Candeira S/N
tel: 981520829, fax: 981520829
josealbert.pazos@rai.usc.es, eljpet@usc.es, joserios@usc.es, manel.cotos@usc.es

Resumen

En este artículo se presenta el análisis, diseño e implementación de un servicio web que permite la simulación de la propagación de incendios forestales. La interfaz del servicio soporta los estándares definidos por el World Wide Web Consortium (W3C), manteniendo al mismo tiempo características similares a la mayoría de servicios propuestos por el Open Geospatial Consortium (OGC), utilizando en la medida de lo posible interfaces y formatos definidos por el mismo. Además de la adecuación a estándares actuales, a la hora de diseñar el servicio se han tenido en cuenta requerimientos de evolución futura y restricciones de coste por licencia. Estos requerimientos y restricciones han guiado la elección de un entorno de programación basado en el lenguaje Java y la utilización generalizada de herramientas de código abierto. Un último requerimiento importante que merece la pena resaltar es el establecimiento de una interfaz de comunicación asíncrona entre el servicio desarrollado y sus clientes, que permite a los mismos la ejecución de otras tareas en paralelo con los procedimientos de simulación ejecutados en el servicio.

Palabras clave: Simulación Incendios, Servicios Web, Cobertura Geográfica, WCS

1 Introducción

Los actuales escenarios de gestión de emergencias naturales ponen de manifiesto la necesidad de sistemas de información que sean capaces de ofrecer a los tomadores de decisiones toda aquella información que sea relevante, en el momento preciso y en el lugar en el que se encuentren. En el contexto medioambiental gallego, resulta especialmente significativa la devastación causada año tras año por los incendios forestales. Un buen ejemplo de esto es la ola de incendios que asoló Galicia entre los días 4 y 14 de agosto de 2006, dejando una fatal estadística de casi 2.000 incendios, cerca de 80.000 hectáreas de superficie forestal afectada y cuatro víctimas mortales.

Los avances producidos en los últimos años en el campo de las tecnologías de la información y las comunicaciones proporcionan medios precisos para conseguir que todos los actores implicados en

la extinción de incendios forestales dispongan en tiempo y forma de información de gran calidad necesaria para tomar sus decisiones. Más concretamente, en el área de los Sistemas de Información Geográfica (SIG) [1] se dispone en la actualidad de herramientas que facilitan la gestión de información referente tanto a *entidades geográficas* como a *coberturas geográficas*. La gestión de propiedades alfanuméricas y geométricas de las *entidades geográficas* es muy común en un gran número de aplicaciones SIG actuales, sin embargo, la gestión de *coberturas geográficas* es vital en la lucha contra incendios forestales. Una *cobertura geográfica* es un conjunto de funciones con un dominio geográfico común (parte de la superficie terrestre) y cada una de ellas con un rango alfanumérico determinado. En general, el dominio geográfico de una cobertura está determinado por alguna forma geométrica, que puede ser una superficie o una línea. Cada función de una cobertura se llama *banda* o *atributo* de la misma. Un ejemplo de cobertura geográfica muy importante en nuestro contexto es una cobertura de datos meteorológicos, con bandas que asignan a cada punto del espacio la temperatura, dirección del viento, velocidad del viento, humedad relativa, etc. Otro ejemplo de cobertura geográfica importante sería una cobertura de terreno, con bandas que asignan a cada punto del espacio la elevación, la pendiente, el aspecto, etc. Un último ejemplo muy importante sería una cobertura de tipos de vegetación que asigna a cada punto del espacio un índice que determina el tipo de combustible vegetal presente en ese punto (modelo de combustible).

Respecto a la propagación de incendios forestales, el desarrollo durante los años 70 y 80 de diferentes modelos de predicción del comportamiento del fuego dio lugar a la aparición de herramientas informáticas de libre disposición que simulan dicha propagación. En cuanto a los modelos de comportamiento, los más sencillos simulan la propagación del fuego como un proceso de contagio entre puntos vecinos del espacio geográfico en cuestión. La velocidad de propagación entre dos puntos dependerá de los valores de algunas bandas de coberturas implicadas, como las meteorológicas, de terreno y de tipos de vegetación mencionadas arriba. Modelos de propagación de mayor complejidad, y también de mayores prestaciones, son los basados en el principio de Huygens para simular el crecimiento como una onda elíptica. El incendio en este caso se propaga en un número finito de pasos temporales, usando en cada paso puntos del frente del incendio como fuentes independientes de pequeñas ondas elípticas. Estas elipses envuelven el perímetro original del fuego para establecer el nuevo frente en cada iteración; la forma y dirección de cada elipse viene determinada por condiciones de viento y pendiente, mientras que su tamaño está condicionado por las características de la vegetación.

En cuanto al software de simulación disponible de forma libre para la comunidad de interesados en el tema, son de especial relevancia los productos publicados por el SEM (Systems for Environmental Management) [2], que incluyen tanto aplicaciones finales como componentes de desarrollo. El principal inconveniente de las aplicaciones finales existentes es que son soluciones cerradas y monolíticas, incapaces de interoperar con otros sistemas de información tanto para acceder a sus fuentes de datos de forma remota como para servir los resultados a través de interfaces estándar. En conclusión, dichas aplicaciones non son incorporables directamente en una Infraestructura de Datos Espaciales (IDE) existente que disponga de la información necesaria para el proceso de simulación.

En este artículo se presenta el análisis, diseño e implementación de un servicio web de la capa de procesamiento de la arquitectura propuesta por el Open Geospatial Consortium (OGC) que permite la generación de datos simulados relativos a la propagación de incendios forestales. Las características del servicio desarrollado pueden resumirse como sigue:

- El cliente especifica las fuentes de datos de las que el servicio obtiene las entradas. Estas fuentes de datos son proporcionadas mediante la interfaz estándar del OGC Web Coverage Service (WCS) [3].

- Los datos geográficos de salida del servicio se proporcionan también a través de la interfaz WCS del OGC.
- La interfaz está basada en estándares del World Wide Web Consortium (W3C). En concreto se utiliza Simple Object Access Protocol (SOAP) [4] para la transferencia de datos y Web Services Description Language (WSDL) [5] para la descripción de la interfaz.
- La comunicación entre cliente y servidor es asíncrona. Después de iniciada la simulación, el cliente podrá consultar el estado de la misma para poder saber cuando los datos de salida están ya disponibles para descarga (a través del WCS).
- El desarrollo está basado en herramientas libres y es fácilmente transportable entre plataformas distintas.
- El diseño del servicio es flexible, permitiendo la fácil incorporación de nuevos modelos de propagación, nuevas entradas y nuevas salidas.

El resto del artículo se organiza como sigue. En la Sección 2 se proporciona una descripción breve de los requesitos impuestos al sistema. El diseño del sistema, que incluye la aquitectura de componentes utilizada y la descripción de la comunicación entre los mismos, se describe en la Sección 3. La Sección 4 proporciona detalles relacionados con la implementación, incluyendo detalles sobre la interfaz del servicio y también de forma informal cuestiones relacionadas con el algoritmo de propagación del fuego implementado. La Sección 5 concluye el artículo y describe algunas líneas de trabajo futuro.

2 Requisitos del sistema

En esta sección se resume de forma breve el análisis de los requisitos del sistema desarrollado. Estos requisitos se han clasificado en dos grandes grupos: i) requisitos relacionados con la funcionalidad del sistema (*requisitos funcionales*) y ii) otros requisitos, relacionados con el diseño del propio sistema, su interfaz, calidad y evolución futura. En cuanto a los *requisitos funcionales*, de forma genérica, el sistema permite la simulación del comportamiento de un incendio forestal a partir de datos suministrados por el usuario. Una descripción más detallada de estos requisitos se proporciona abajo en esta sección. Los demás requisitos pueden resumirse como sigue:

- *Comunicación asíncrona*: Debido al alto coste computacional de los algoritmos de propagación se ha requerido la implementación de un servicio web de simulación con una interfaz de comunicación asíncrona con los clientes. De esta forma los clientes de este servicio no están obligados a detener su ejecución esperando por el resultado de una simulación, sino que una vez iniciada la misma pueden continuar su ejecución con otras tareas, pudiendo en todo momento obtener información sobre su evolución.
- *Interfaces OGC*: Los datos geográficos se sirven del cliente al servidor y del servidor al cliente siempre mediante interfaces web estándar del OGC. En concreto, se utilizó la interfaz WCS para la transferencia de las coberturas de entrada y salida y se hizo uso de la especificación Web Map Service (WMS) [6] para la visualización de entradas y salidas. En la medida de lo posible se utilizaron también elementos del formato Geography Markup Language (GML) [7].
- *Evolución futura*: El sistema se ha desarrollado teniendo en cuenta restricciones de escalabilidad, flexibilidad y funcionamiento multiplataforma. En concreto, el sistema permite la incorporación de nuevas fuentes de datos, siempre que sus servidores utilicen estándares OGC. El conjunto de algoritmos de propagación no es cerrado, sino que puede ampliarse (hay que destacar aquí que el objetivo del proyecto no es el desarrollo de nuevos

algoritmos de propagación). Finalmente, la utilización de tecnología basada en Java permite la migración fácil entre plataformas basadas en Microsoft Windows y plataformas basadas en Linux.

- *Interfaz de usuario final*: Para ilustrar el funcionamiento de servicio web desarrollado se ha desarrollado también una aplicación web final de propósito general. Esta aplicación se ha desarrollado en base a requerimientos de facilidad de uso y sencillez. La interfaz propuesta es intuitiva y su utilización no requiere ningún tipo de formación.
- *Licencias*: A lo largo de todo el desarrollo del servicio se han utilizado tecnologías sin coste por licencia asociado. En la medida de lo posible se han utilizado herramientas libres y de código abierto.

Se proporcionan en el resto de esta sección detalles sobre los *requerimientos funcionales* del sistema. El diagrama de casos de uso vinculado a estos requerimientos puede verse en la Figura 1, y la descripción breve de cada uno de estos casos se proporciona a continuación.

Iniciar Simulación: Un cliente del servicio web pueden en cualquier momento iniciar un nuevo proceso de simulación. El cliente debe de especificarle al servicio información sobre cada una de las fuentes de datos que se van a utilizar, de manera que el servicio pueda recuperarlas a través de interfaces estándar de OGC. Además, el cliente tiene que especificar otros parámetros necesarios para la simulación (puntos de ignición, tiempo de simulación, resolución de salida, etc.). Una vez iniciada la simulación el servicio devuelve al cliente un identificador de la misma que éste utilizará en futuras peticiones.

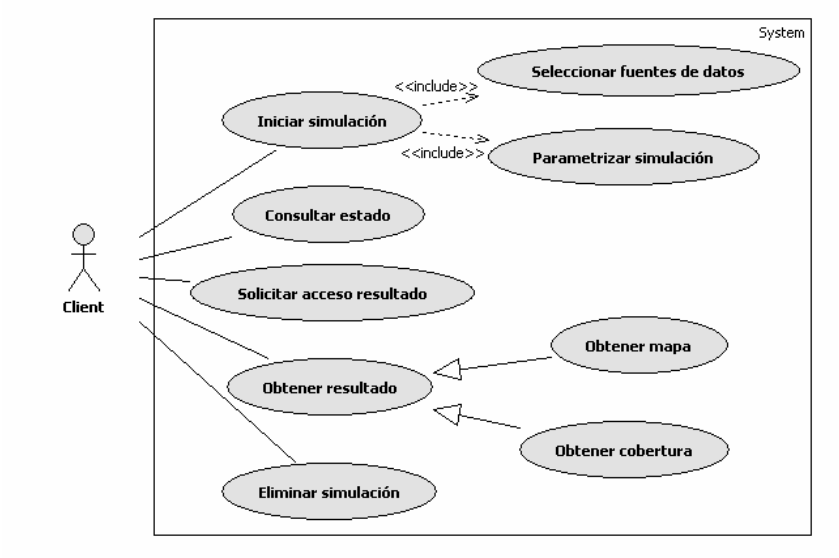


Figura 1. Diagrama de casos de uso.

Consultar estado: Para poder consultar el estado en el que se encuentra una simulación, ésta debe de haber sido antes iniciada y el cliente debe de poseer ya un identificador de la misma.

Solicitar acceso resultado: Una vez verificada la finalización de una simulación y utilizando su identificador, el cliente puede solicitar al servicio información sobre cómo puede obtener los datos de salida. Estor datos serán servidos a través de la web mediante interfaces OGC, como se dice en el caso de uso siguiente.

Obtener resultado: El resultado de una simulación se obtiene a través de la web mediante interfaces

de servicio OGC. En concreto, cada una de las partes del resultado puede obtenerse en forma de cobertura geográfica a través de una interfaz WCS o en forma de mapa a través de una interfaz WMS. Estos resultados pueden obtenerse cuantas veces sea necesario.

Eliminar simulación: El cliente del servicio puede en cualquier momento solicitar que se borre toda la información almacenada sobre una determinada simulación, incluyendo su resultado. Para esto necesitará conocer su identificador. El administrador del servicio podrá decidir también que sea necesario borrar datos sobre simulaciones a los que no se haya accedido recientemente.

3 Diseño global

En base al análisis de requerimientos de la sección anterior se presentan en ésta algunas cuestiones relacionadas con el diseño del sistema. En concreto, la arquitectura de componentes del sistema se describe breve e informalmente en la sub-sección 3.1. La sección 3.2 se dedica a la descripción de la secuencia de funcionamiento en un caso típico de uso correcto del mismo (no se incluyen detalles relacionados con la gestión de casos de error).

3.1 Arquitectura del sistema

La arquitectura de componentes del sistema se presenta de forma gráfica en el diagrama UML de la Figura 2. En esta sección se incluye una descripción breve de cada uno de los componentes de esta arquitectura. Como puede verse en la figura, la arquitectura se descompone en tres grandes bloques que se comunican, en general, a través de la Internet (en concreto mediante el protocolo HTTP).

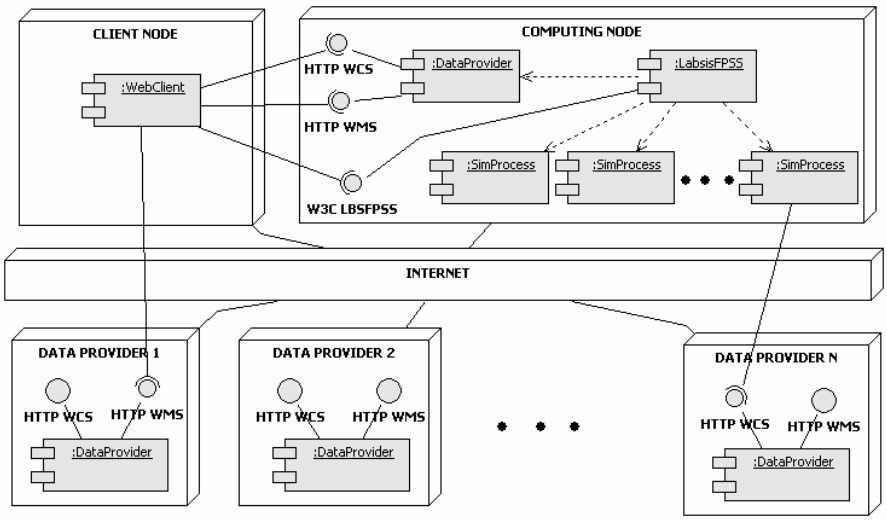


Figura 2. Arquitectura basada en servicios web

En un primer bloque (parte inferior de la figura) se encuentran uno o varios proveedores de datos. La misión de estos proveedores es almacenar y servir los datos de entrada que necesitan los procesos de simulación. Estos datos se sirven tanto en forma de coberturas a través de una interfaz WCS como en forma de mapas a través de una interfaz WMS. Como puede verse en la figura, los clientes del sistema (en concreto el cliente web de pruebas desarrollado) pueden utilizar la interfaz WMS para visualizar los datos de entrada, mientras que la interfaz WCS la utilizan los procesos de simulación para acceder a los valores de cada parámetro en cada localización del sub-espacio

geográfico de trabajo.

El segundo bloque (parte superior derecha de la figura) contiene los componentes principales del sistema, es decir, los componentes del servicio web que implementa los procesos de simulación de incendios. El componente *LabsisFPSS* (Labsis Fire Propagation Simulation Service) recibe las peticiones de simulación de los clientes a través de su interfaz W3C. Cada proceso de simulación es atendido por una instancia del componente *SimProcess*. Este componente accede a los datos de entrada suministrados por los proveedores correspondientes y genera los datos de salida de la simulación. Estos datos de salida se sirven mediante el componente *DataProvider* a través de sus interfaces OGC, tanto WCS como WMS.

El último bloque de la arquitectura (parte superior izquierda de la figura) contiene los componentes que actúan como clientes del sistema. En este caso se ha desarrollado una pequeña aplicación web de pruebas que permite ejecutar procesos de simulación. Este cliente permite seleccionar los proveedores de datos que se van a utilizar, y visualizar sus capas mediante la interfaz WMS. Utiliza la interfaz W3C del servicio web para gestionar los procesos de simulación y permite visualizar las salidas de dichos procesos mediante el uso de la interfaz WMS del DataProvider del servicio de simulación.

3.2 Descripción de la comunicación entre componentes

Una vez vista en la sub-sección anterior la arquitectura global del sistema se ilustra a continuación en más detalle la comunicación entre los distintos componentes de la misma para resolver un caso típico de funcionamiento. La secuencia de intercambio de mensajes entre los distintos componentes se representa de forma gráfica mediante el diagrama de secuencia UML de la Figura 3. Detalles concretos sobre la comunicación entre los componentes se describen a continuación.

Como puede verse en la figura, la secuencia se inicia con el envío de una petición *getCapabilities* entre el cliente web desarrollado (*WebClient*) y el componente principal del servicio de simulación (*LabsisFPSS*). Ésta es una operación de la interfaz W3C del servicio de simulación que permite al cliente recuperar metadatos sobre el propio servicio, al estilo de todas las interfaces web propuestas por el OGC.

A continuación el cliente envía una petición de inicio de simulación *initSimulation*. Esta petición incluirá los parámetros necesarios para la propia simulación, que incluirán las direcciones donde el servicio puede encontrar cada uno de los datos de entrada. El componente *LabsisFPSS* crea una nueva instancia del *SimProcess* pasándole todos los parámetros que ha recibido del cliente y delega en el mismo la responsabilidad de realizar la simulación. Una vez hecho esto devuelve al cliente un identificador de simulación (sim1 en este caso).

Para realizar la simulación el componente *SimProcess* empieza por la petición de los datos de entrada. Para esto realiza peticiones *getCoverage* a los proveedores de datos a través de sus interfaces WCS. En paralelo el cliente puede realizar peticiones *getSimulationState* al componente *LabsisFPSS* para seguir la evolución del estado de la simulación. Las peticiones que se reciben mientras se obtienen los datos de entrada son devueltas con el estado *ObtainingData*, como puede observarse en la figura.

Una vez obtenidos los datos de entrada el proceso continúa con la ejecución del proceso de simulación (*doSimulation*). En este momento la simulación se encuentra en estado *ProcessingData*. Una vez terminado el proceso de simulación se prepara la salida *outElaboration* y se configura el componente *DataProvider* incrustado en el servicio para que pueda servir los datos. Durante este

tiempo la simulación se encuentra en estado *PreparingOutput*. Una vez terminada la preparación de la salida la simulación pasa a estado *Completed* y el cliente ya puede solicitar acceso a los datos de salida. Como puede verse en la figura, en cualquier momento (incluso en medio de una simulación), el componente *LabsisFPSS* puede recibir una nueva petición de inicio de simulación *initSimulation*, que será atendida en paralelo por un nuevo *SimProcess*. En este caso se muestra en la figura una petición de inicio de simulación realizada por un cliente externo que no es nuestra aplicación web cliente.

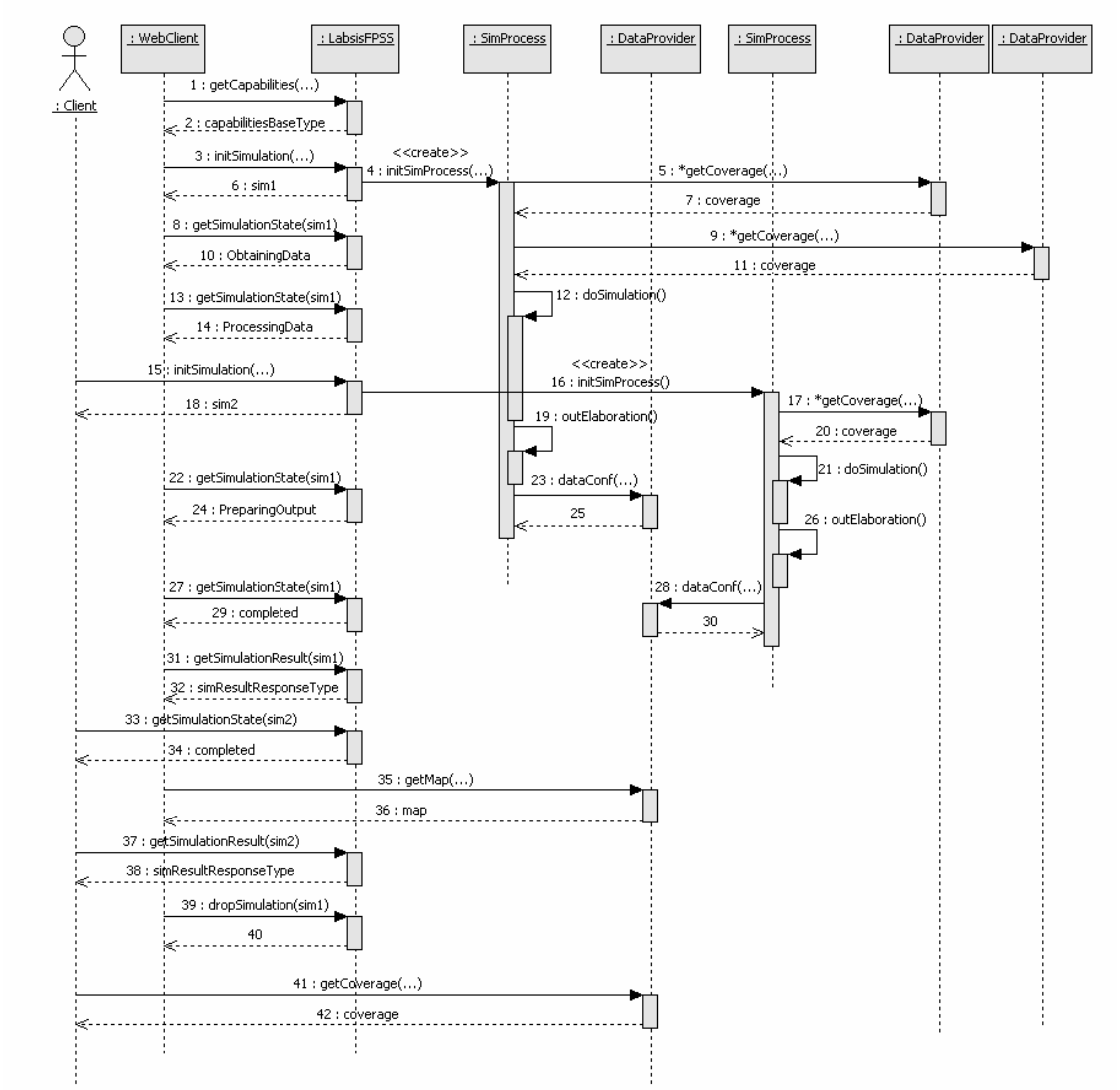


Figura 3. Diagrama de secuencia.

Una vez el cliente es informado de que la simulación ha terminado (estado *Completed*), éste puede solicitar acceso a los datos de salida mediante una petición *getSimulationResult*. La respuesta a esta petición incluye información sobre el *DataProvider* incrustado que puede utilizar el cliente para acceder a la salida. Este acceso se realizará mediante peticiones *getMap* de la interfaz WMS, mediante peticiones *getCoverage* de la interfaz WCS o mediante ambas. En el caso del cliente web desarrollado, dado que sólo permite visualizar la salida, será suficiente con peticiones *getMap*. Estas peticiones podrán seguir realizándose mientras los datos de salida estén disponibles en el

DataProvider.

El cliente tendrá la capacidad de eliminar estos resultados y todos los datos generados por una simulación mediante una petición *dropSimulation* al componente *LabsisFPSS*. Por razones obvias de mantenimiento, el administrador del servicio puede decidir la eliminación de simulaciones cuyos datos de salida no hayan sido utilizados recientemente.

4 Detalles de implementación

Una vez vista una visión general del diseño del sistema en la sección anterior, se presentan en ésta algunas consideraciones relacionadas con detalles de implementación de los distintos componentes. Respecto a la plataforma y herramientas de implementación utilizadas, su elección se ha basado en requerimientos impuestos al sistema (ver Sección 2). Algunas características relevantes que han guiado dicha elección pueden resumirse como sigue:

- *Plataforma base y entorno de programación*: La implementación de un sistema multiplataforma es un requisito ya visto en la Sección 2. La utilización de un entorno de programación Java (JDK 1.5) y de herramientas con versiones disponibles tanto para sistemas Windows como para distribuciones de Linux hace que este requerimiento se haya podido cumplir.
- *Servicios web*: La programación Java y la necesidad de incluir funcionalidad en web, y las restricciones de costes por licencias determinó la elección de herramientas derivadas de proyectos Apache (http://www.apache.org/), sin duda las más utilizadas en la actualidad en entornos de software libre. La programación de Servlets y páginas JSP se apoyaron en el contenedor Tomcat de Apache (http://tomcat.apache.org/). Para la implementación del componente *LabsisFPSS* como un servicio web se optó por la herramienta de apache AXIS (http://ws.apache.org/axis/), que simplifica el acceso y creación de servicios web W3C en entornos Java, evitando al programador el tener que manejar detalles relacionados con la codificación SOAP de las peticiones.
- *Proveedores de datos OGC*: Para la implementación de proveedores de datos con interfaces OGC (WMS y WCS) se consideraron tres herramientas bien conocidas de software libre: MapServer [8], GeoServer [9] y deegree [10]. Dado que la gran mayoría de los datos gestionados por el sistema son de tipo raster, se optó finalmente por una solución basada en la herramienta MapServer. Hay que resaltar sin embargo que el componente *LabsisFPSS* es capaz de obtener datos de entrada de cualquier proveedor que implemente la interfaz estándar WCS de OGC y que de forma similar la aplicación web desarrollada es capaz de visualizar datos de cualquier proveedor que implemente la especificación WMS de OGC.

Más detalles sobre la implementación del componente *LabsisFPSS* se describen en las siguientes sub-secciones. En concreto, la Sub-sección 4.2 presenta algunos detalles sobre la interfaz del componente, mientras que detalles relacionados con el algoritmo de simulación de la propagación del fuego utilizado se incluyen en la Sub-sección 4.3.

4.2 Interfaz W3C del Servicio

Se ha visto ya en la Sub-sección 3.1 que el componente principal del sistema *LabsisFPSS* proporciona su funcionalidad en forma de servicio web con interfaz W3C. Las operaciones de dicha interfaz que proporcionan la posibilidad de comunicación asíncrona entre el componente y sus

clientes se han visto en la Sub-sección 3.2 en el contexto de una secuencia de funcionamiento típica. Estas operaciones son: *getCapabilities*, *initSimulation*, *getSimulationState*, *getSimulationResult* y *dropSimulation*.

La operación *getCapabilities* es común a todas las interfaces de servicios web del OGC. En el caso de este servicio, el resultado obtenido es directamente un elemento del tipo *ows:CapabilitiesBaseType* definido en [11]. La operación *getSimulationState* devuelve con un tipo de dato string el estado en el que se encuentra la simulación cuyo identificador se pasa como parámetro. La operación *dropSimulation* elimina del servidor la simulación cuyo identificador se pasa como parámetro. La interfaz de las restantes dos operaciones, descrita con más detalle a continuación, es más compleja.

El trozo del código WSDL que define la interfaz de la operación *initSimulation*, junto con el XML Schema que define alguno de los tipos más importantes de los incluidos en dicha interfaz se proporciona en la Figura 4. Los parámetros de entrada de esta operación, tal y como se muestra en la figura son:

- *service*: Es un parámetro típico de todo servicio OGC. El valor utilizado para este servicio es "FPSS".
- *version*: Versión de la interfaz utilizada (1.0 en este caso).
- *ignition*: Localización o localizaciones en las que se inicia el fuego. Como puede verse en la definición del tipo *IgnitionType* en la figura, el valor de este parámetro es un punto (elemento *firstPosition* especificado en formato GML). Opcionalmente, pueden incluirse más puntos (elemento *nextPositions*). Cada uno de estos nuevos puntos de ignición tendrá asociado un tiempo de retardo desde el inicio de la simulación.
- *simulationTime*: Duración de la simulación en minutos.
- *srsName*: Identificador del sistema de referencia de coordenadas utilizado. Este identificador se utilizará en las peticiones WCS y en la generación y configuración de las salidas del servicio.
- *BBox*: Superficie rectangular especificada en GML que define el área geográfica sobre la cual el fuego puede expandirse. Este parámetro se utilizará en las peticiones WCS a los proveedores de datos.
- *resolution*: Resolución en unidades del sistema de referencia especificado en el parámetro srsName.
- *windDirection, windSpeed, slope, aspect, fuelModel*: Dirección y velocidad del viento, pendiente del terreno, orientación del terreno y modelo de combustible, respectivamente. Cada uno de estos parámetros puede ser considerado o bien constante en toda el área de acción del fuego o bien variablemente determinado por una cobertura. En la práctica, sólo puede tener sentido considerar como constantes la dirección y velocidad del viento y ninguno más. El significado de cada uno de estos parámetros se proporciona con más detalle en la siguiente sub-sección. Para su especificación en la interfaz del servicio, tal y como se muestra en la figura, se utiliza el tipo de dato *SimulationParameter*, definido específicamente para este servicio. Un parámetro de simulación puede ser o una constante de tipo *gml:MeasureType* (elemento *constant* en la figura), lo que significa que el parámetro se considera como de valor constante en toda el área de acción del fuego, o una cobertura, especificada mediante un elemento de tipo *lbsfss*:*remoteDataSourceType*, que define el valor del parámetro en cada localización del área de acción de fuego. Un elemento de tipo *lbsfss*:*remoteDataSourceType* proporciona información para localizar la cobertura tanto en un WCS como en un WMS. En concreto especifica la dirección del servicio WCS que sirve la cobertura (elemento *coverageServiceLocator*), el nombre de la cobertura (elemento *coverageName*), la dirección del servicio WMS que permite visualizar la

cobertura (elemento *mapServiceLocator*) y el nombre de la capa dentro del WMS (elemento *layerName*).

```
<message name="InitSimulationRequest">
    <part name="service"
          type="xsd:string"/>
    <part name="version"
          type="xsd:string"/>
    <part name="ignition"
          type="lbsfss:IgnitionType"/>
    <part name="simulationTime"
          type="xsd:duration"/>
    <part name="srsName"
          type="anyURI"/>
    <part name="BBox"
          type="gml:EnvelopeType"/>
    <part name="resolution"
          type="gml:MeasureType"/>
    <part name="windDirection"
          type="lbsfss:SimulationParameter"/>
    <part name="windSpeed"
          type="lbsfss:SimulationParameter"/>
    <part name="slope"
          type="lbsfss:SimulationParameter"/>
    <part name="aspect"
          type="lbsfss:SimulationParameter"/>
    <part name="fuelModel"
          type="lbsfss:SimulationParameter"/>
 </message>

 <message name="InitSimulationResponse">
    <part name="response"
          type="lbsfss:InitSimulation
                              ResponseType"/>
 </message>
```

```
<complexType name="IgnitionType">
 <sequence>
  <element name="firstPosition"
           type="gml:DirectPositionType"/>
  <element name="nextPositions"
           Type="lbsfss:NextPositionType"
                 minOccurs="0"
                 maxOccurs="unbounded"/>
 </sequence>
</complexType>

<complexType name="NextPositionType">
 <sequence>
  <element name="position"
           type="gml:DirectPositionType"/>
  <element name="timeOffset"
           type="duration"/>
 </sequence>
</complexType>

<complexType name="SimulationParameter">
 <choice>
  <element name="constant"
           type="gml:MeasureType"/>
  <element name="remoteDataSource"
       type="lbsfss:remoteDataSourceType"/>
 </choice>
</complexType>

<complexType name="remoteDataSourceType">
 <sequence>
  <element name="coverageServiceLocator"
           type="ows:OnlineResourceType"/>
  <element name="coverageName"
           type="string"/>
  <element name="mapServiceLocator"
           type="ows:OnlineResourceType"/>
  <element name="layerName"
           type="string"/>
 </sequence>
</complexType>
```

Figura 4. Sección del WSDL para la interfaz de la operación initSimulation.

La respuesta a la operación *initSimulation*, codificada en un elemento de tipo *lbsfss:InitSimulationResponseType* contiene o un mensaje de error o un identificador generado por el servicio para la simulación recién iniciada. Este identificador será utilizado por el cliente en futuras comunicaciones con el servicio. Una de estas posibles futuras comunicaciones es la petición de la operación *getSimulationResult*, mediante la cual el cliente solicita información sobre dónde obtener los datos de salida de una simulación en curso. El pedazo de código WSDL que define los parámetros de la interfaz de esta operación se proporciona en la Figura 5, junto con el XML Schema de algunos tipos necesarios. En concreto, puede observarse en la figura que la petición incluye los bien conocidos parámetros *service* y *version*, de todas las peticiones, y además un parámetro *simulationID*, en el que el cliente especifica el identificador de la simulación para la cual solicita el resultado.

La respuesta del servicio es un elemento de tipo *GetSimulationResultResponseType*, que contiene o un mensaje de error o información de la salida en un elemento *simulationResult*. Éste elemento proporciona información sobre como acceder a cada una de las cuatro salidas de servicio, *timeOfArrival*, *flameLength*, *scorchHeight* y *ByramIntensity*. Cada una de estas salidas se especifica

con un elemento de tipo *lbsfss:SimulationParameter* ya descrito arriba, que determina tanto la localización de los servicios WCS y WMS donde se encuentra la salida como los nombres de cobertura y capa dentro de los mismos. La descripción del significado de cada uno de estos parámetros de salida se proporciona en la siguiente sub-sección.

```
<message name="GetSimulationResultRequest">
 <part name="service"      type="xsd:string"/>
 <part name="version"      type="xsd:string"/>
 <part name="simulationID" type="long"/>
</message>

<message name="GetSimulationResultResponse">
 <part name="response" type="lbsfss:GetSimulationResultResponseType"/>
</message>

<complexType name="GetSimulationResultResponseType">
 <choice>
  <element name="error"            type="string"/>
  <element name="simulationResult" type="lbsfss:SimulationResultType"/>
 </choice>
</complexType>

<complexType name="SimulationResultType">
 <sequence>
  <element name="timeOfArrival"  type="lbsfss:SimulationParameter"/>
  <element name="flameLength"    type="lbsfss:SimulationParameter"/>
  <element name="scorchHeight"   type="lbsfss:SimulationParameter"/>
  <element name="ByramIntensity" type="lbsfss:SimulationParameter"/>
 </sequence>
</complexType>
```

Figura 5. Sección del WSDL para la interfaz de la operación getSimulationResult.

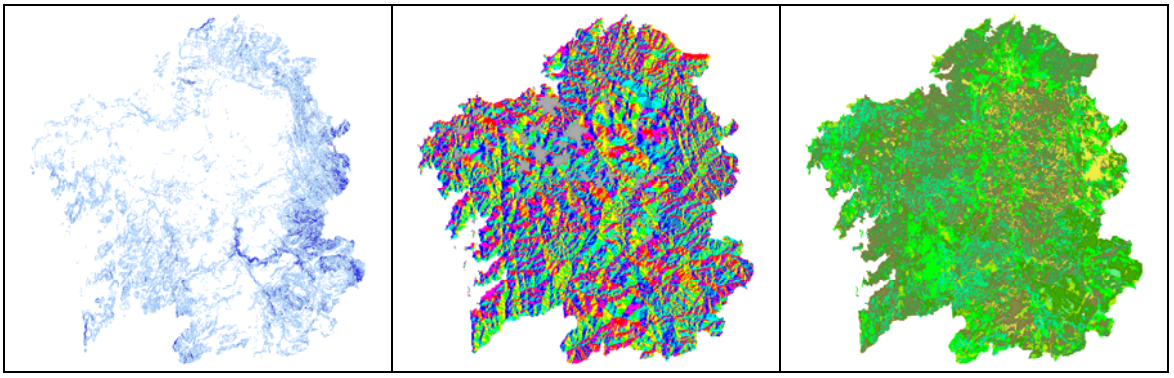
4.3 Algoritmo de propagación

En la introducción se hace referencia a dos grandes familias de algoritmos de predicción del comportamiento del fuego. La alternativa más sencilla la constituyen los modelos celulares que simulan la propagación como un proceso de contagio entre vecinos. Dichos modelos iteran para obtener tiempos de ignición en celdas regularmente espaciadas en base a las condiciones meteorológicas, del terreno y de la vegetación. Si bien el principal defecto de esta aproximación es la distorsión resultante en la forma del fuego cuando se utiliza un número reducido de vecinos, se ha optado por esta vía por ser más simple de programar y menos costosa desde el punto de vista computacional.

Respecto al software libre proporcionado por el SEM, al margen de utilidades de escritorio de gran popularidad como BEHAVE o Farsite, encontraremos dos herramientas destinadas a los usuarios interesados en desarrollar sus propios componentes de simulación: la librería fireLib, basada en el modelo de contagio celular anteriormente referido, y su sucesor, mucho más potente y complejo, el kit de desarrollo Fire Behavior SDK. Dada la sencillez de la primera, se ha considerado oportuno basar en ella la construcción del componente de simulación, toda vez que el presente trabajo se centra en la especificación de la interfaz del servicio de simulación de incendios y no en su implementación.

La librería fireLib proporciona una API sencilla y optimizada que permite desarrollar aplicaciones capaces de determinar el ritmo de propagación o la intensidad de un incendio forestal. Codificada en ANSI C, resulta factible implementar con sólo 4 de sus 13 funciones un simulador simple, eficiente y funcional. Nuestro componente de simulación, que ejecuta un algoritmo simple de contagio a los 8 vecinos más próximos introduciendo errores geométricos significativos, admite las siguientes entradas:

- Coberturas derivadas de un Modelo Digital del Terreno (MDT): Pendiente en tanto por ciento y orientación en grados medidos desde el norte en el sentido de las agujas del reloj.
- Cobertura de modelos de combustible: clasificación del suelo forestal en 13 modelos estándar definidos por el National Forest Fire Laboratory (NFFL) que recogen características como la naturaleza de la vegetación (herbazal, matorral, arbolado y restos), o su altura.
- Velocidad y dirección del viento: medidos en km/h y grados desde el norte en el sentido de las agujas del reloj, respectivamente. Para ambos parámetros, se admiten valores constantes o coberturas.
- Coordenadas de uno o varios focos en el sistema de referencia de coordenadas especificado.
- Tiempo de ignición para cada foco en minutos.
- Tiempo de simulación en minutos.



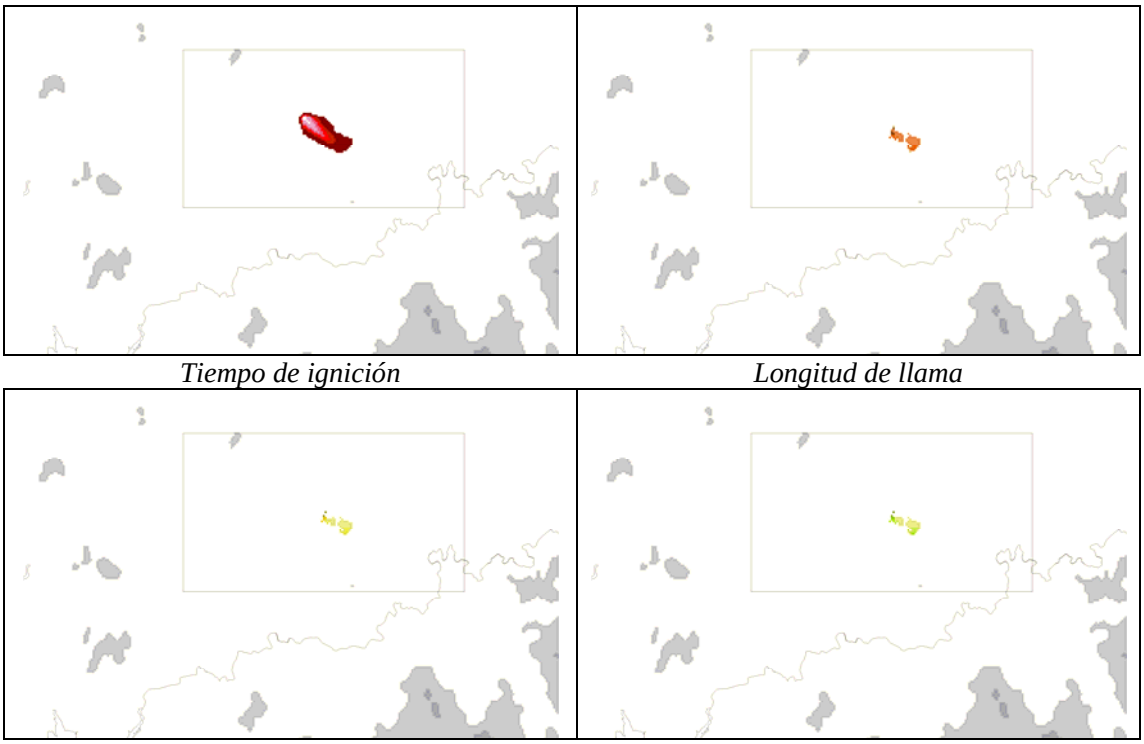
Pendiente *Orientación* *Combustible*

Figura 6. Ejemplos de coberturas de entrada para una simulación.

En la figura 6 se muestra el aspecto de algunas de las coberturas que podrían suministrarse como entradas. Como salida de una simulación, el usuario obtiene a discreción hasta cuatro coberturas diferentes que almacenan distinta información:

- Tiempo de ignición (*timeOfArrival*) en minutos: refleja el tiempo de llegada del fuego a cada celda de la cobertura.
- Longitud de las llamas (*flameLength*) en metros: indica la longitud de la llama desde su base hasta su extremo.
- Altura de chamuscamiento (*scorchHeight*) en metros: recoge la altura máxima a la que se observará quemada la vegetación (no tiene por qué coincidir con la longitud de las llamas, ya que estas se pueden inclinar por la acción del viento).
- Intensidad lineal del fuego (*ByramIntensity*) en KW/m: como densidad lineal de la potencia entregada por el fuego, da una medida de la energía que transporta en su frente.

La figura 7 refleja el resultado de una simulación que utilizó las coberturas de la Figura 6 como entradas y un viento constante del noroeste. Los tonos más claros corresponden a valores más bajos de cada variable.



Tiempo de ignición *Longitud de llama*

Altura de chamuscamiento *Intensidad*

Figura 7. Ejemplos de coberturas de salida para una simulación.

5 Conclusiones y trabajo futuro

En este artículo se ha presentado el análisis, diseño e implementación de un servicio web que proporciona funcionalidad para la simulación de la propagación de un incendio forestal. La arquitectura basada en servicios del sistema hace uso de los estándares propuestos por el W3C y de las recomendaciones y estándares del OGC. La interfaz del servicio permite una comunicación asíncrona con sus clientes y ha sido diseñada de forma que pueda absorber de forma sencilla futuras ampliaciones, tanto en el número y tipo de las entradas y salidas como en los algoritmos de propagación utilizados. Para el desarrollo del sistema se ha utilizado una metodología basada en un desarrollo incremental a base de iteraciones, realizando pruebas intensivas a lo largo de cada una de ellas. Respecto a las líneas de trabajo futuro se puede destacar la incorporación de modelos de propagación más complejos, incluyendo algoritmos más sofisticados y condiciones de entrada nuevas, como por ejemplo información meteorológica más precisa.

Referencias

[1] P. Rigaux, M. Scholl, A. Voisard, Spatial Databases with application to gis. Morgan Kaufmann, 2002.

[2] http://fire.org

[3] OGC, OpenGIS Web Coverage Service Implementation Specification, version 1.1, 2006, Descargado de http://www.opengeospatial.org/

[4] W3C, Simple Object Access Protocol (SOAP) 1.1, 2000, Descargado de http://www.w3.org/TR/soap/.

[5] W3C, Web Services Description Language (WSDL) 1.1, 2001, Descargado de

http://www.w3.org/TR/wsdl
[6] OGC, OpenGIS Web Map Server Implementation Specification, version 1.3.0, 2006, Descargado de http://www.opengeospatial.org/
[7] ISO, Geographic information - Geography Markup Language (GML), ISO/TC 211/WG 4/PT 19136, Committee Draft, 2004.
[8] MapServer, http://mapserver.gis.umn.edu/
[9] GeoServer, http://docs.codehaus.org/display/GEOS/Home
[10] deegree, http://deegree.sourceforge.net/
[11] OGC, OpenGIS Web Service Common Implementation Specification, version 1.1.0, 2007. Descargado de http://www.opengeospatial.org/