Geomática aplicada a un Sistema de Alerta Temprana.

Peralta GS, De Elia EA, Lanfri M, Porcasi X, Lanfri S, Frutos N, Rotela C, Scavuzzo M, Lanfri S.

Instituto Mario Gulich, Comisión Nacional de Investigaciones Espaciales, Córdoba, Argentina

Resumen

En el presente trabajo se plantea una Infraestructura Informática para dar soporte a la prevención y el control estratégico del vector del dengue en Argentina por parte del Ministerio de Salud de la Nación. La misma es parte de un complejo Sistema de Alerta Temprana (SAT) y es llevado a cabo por la Comisión Nacional de Actividades Espaciales (CONAE) bajo los estándares de la European Spatial Agency (ESA). La arquitectura, diseño, metodología y codificación pretenden ser componentes re-utilizables en cualquier infraestructura informática de soporte a SATs. En su desarrollo se utiliza Open Source Software (OSS) y Patrones de Diseño (Design Patterns) garantizando una herramienta tecnológica además de re-utilizable, flexible, mantenible y robusta. En este documento se describen los requerimientos establecidos por el Ministerio de Salud de la Nación y se plantea la arquitectura y diseño del sistema a partir de los mismos. Además, se realiza un análisis de las tecnologías OSS integradas en el desarrollo y la codificación. Finalmente se describe la funcionalidad obtenida y se muestra la infraestructura con un caso en particular.

Palabras Claves

Geomática, Sistemas de Alerta Temprana, Open Source Software, Geomajas, GeoServer

1 Introducción

1.1 Motivación

El Dengue es en la actualidad una de las enfermedades transmitidas por vectores de mayor prevalencia en el continente Sudamericano [19]. Múltiples factores ambientales tanto de carácter biofísico como social constituyen una compleja trama que condiciona o determina la proliferación del vector-enfermedad [23][10][26]. Actualmente, los límites de distribución de la especie son la frontera norte del país, Santa Rosa (Provincia de La Pampa) al sur [18] y el Departamento de Guaymallén en la provincia de Mendoza [7] (ver figura 1). Para mayor información ver [12].

La Epidemiología Panorámica es una interdisciplina relativamente nueva que involucra la caracterización de áreas eco-geográficas donde las enfermedades se desarrollan [19]. El principal objetivo de esta, es el desarrollo de mapas de riesgo de enfermedades específicas para lograr la formulación de sistemas de alerta temprana (SAT o su denominación en inglés "Early Warning Systems") en salud; mejorando de esta manera las acciones en programas de control y prevención de enfermedades [16]. De esta manera los SAT tienen como objetivo principal el soporte tanto a la toma de decisión como a la prevención de enfermedades altamente relacionadas con el ambiente espacial. La necesidad de un SAT en el caso específico de la Fiebre del Dengue es acentuada aún más debido a la inexistencia por el momento, de vacuna o tratamiento específico para humanos; siendo la prevención y el control estratégico del vector la única solución plausible para la enfermedad.

El Instituto de altos estudios espaciales Mario Gulich (CONAE - Universidad Nacional de Córdoba), tiene como principal objetivo el generar Sistemas de Alerta Temprana en emergencias ambientales usando información Espacial proveniente de distintos sensores remotos. En particular en este trabajo se describe el desarrollo integral de una infraestructura informática de soporte al Sistema de Alerta Temprana en materia de prevención y control estratégico del Dengue: "Sistema de Alerta Temprana y Estratificación de Riesgo de Dengue a escala Nacional y Urbano": SAT/ERDNU.

Figura 1: Distribución Geográfica estimada de Aedes aegypti (gris claro) y Aedes albopictus (gris oscuro) para el año 2008. Extraído de Vezzani y Carbajo (2008).

1.2 Requerimientos

Un resumen de los requerimientos del Ministerio de Salud de la Nación se presenta a continuación. El sistema debe plantearse como multiescala y multifactorial para la generación de productos que permitan:

- Estratificar el riesgo de Dengue a escala nacional mediante un modelo multifactorial (medioambiental, demográfico, eco-epidemiológico) capaz de asignar el riego de circulación viral de Dengue a cada localidad del país.
- Generar cartografía de riesgo de Dengue a escala Urbano, a partir de la caracterización de hábitat preferenciales para el desarrollo del mosquito Aedes aegypti y la dinámica de transmisión del virus.
- Visualizar localidades de riesgo a escala nacional.
- Generar cartografía de áreas de riesgo a escala Urbana a partir de datos espaciales, que complementados con datos de campo, optimicen las acciones de control.
- Integrar la información histórica de la circulación viral de dengue y de otros datos epidemiológicos (flujo poblacional, capacidad de respuesta ante eventos epidémicos, entre otros)
- Actualizar frecuentemente los datos ambientales provenientes de sensores remotos.
- Integrar los datos en una plataforma cartográfica amigable, fácil de utilizar y robusta.

1.2.1 Estratificación del Riesgo del Dengue a escala Nacional: ERDN

Para la estratificación del Riesgo de Dengue a escala Nacional cada localidad está representada en una capa vectorial de puntos (formato .shp), por un par único de coordenadas (latitud, longitud) al que se asociará información alfa-numérica (en formato .dbf) correspondiente a bloques de factores condicionantes del riesgo:

Caracterización macro-ambiental: será determinada principalmente por datos de temperatura de superficie obtenidos del sensor MODIS, por datos de Índice de Vegetación del mismo sensor (o humedad ambiente) y de altura de la localidad. Estos datos, inicialmente en formato raster, se resumirán a un valor por localidad (promedio de píxeles).
Los siguientes datos se derivarán de una planilla de riesgo que deberá ser ingresada por el usuario conforme a un proceso de validaciones automáticas que se describen en 2.3.
Control vectorial: este bloque caracteriza la localidad en función de la periodicidad en que realizan acciones orientadas al control de la densidad de poblaciones del mosquito A. aegypti. Caracterización de la circulación viral: aquí se concentra la información relacionada a circulación autóctona del virus del Dengue en cada localidad. Por ejemplo serotipo circulante.
Riesgo entomológico: este bloque considera la presencia del vector Aedes aegypti en la localidad y otras medidas asociadas a la densidad del mismo. Por ejemplo índices entomológicos como el índice de Breteau.
El riesgo final de circulación de Dengue a escala Nacional para cada localidad es definido a través de la aplicación de un modelo matemático en función a los valores obtenidos en cada bloque. El sistema en su conjunto permitirá flexibilidad tanto para la inclusión de nuevos datos, del Ministerio de Salud, como en la ponderación o peso de cada variable en la asignación de riesgo total.

1.2.2 Estratificación del Riesgo del Dengue a escala Urbano: ERDU

Para el caso de la Estratificación del Riego a escala Urbano y a partir de imágenes satelitales, datos históricos de circulación viral, índices aédicos (LIRAa, ovitrampas, descacharrado, etc.) y cartografía de base en formato vectorial, se estiman anomalías y fluctuaciones espaciales de las variables de relevancia para la estratificación de riesgo de Dengue dentro de cada localidad. Las localidades piloto han sido seleccionadas por solicitud expresa de los referentes epidemiólogos participantes en el diseño de esta herramienta, en base a la disponibilidad de datos de campo y epidemiológicos, a la existencia de eventos epidémicos históricos y recientes, a su localización con respecto a países limítrofes y áreas de circulación viral frecuente.

La tabla 1 muestra un resumen de los requerimientos del usuario (Ministerio de Salud de la Nación).

	ERDN	ERDNAmb	ERDNfact	ERDU	ERDUamb	ERDUFact
	Mapa riesgo de nivel nacional	Subproducto de riesgo nacional ambiental	Estado de los factores no ambientales	Escala Urbana		
Producto	Vector	Raster	Vector	Raster	Raster	Vector
Area	Argentina	Argentina	Argentina	Localidad	Localidad	Localidad
Resolución temporal	semestral	Bi-mensual	semestral	mensual	anual	mensual
Resolución espacial/ escala de mapa	regional	Km	regional	10 - 30 m/cuadras	10 - 30 m/cuadras	10 - 30 m/cuadras
Datos de entrada	RS, planilla estratificación*	RS, planilla estratificación*	RS, planilla estratificación*	Entomológicos ambientales serológicos Servicios etc.	TBD	TBD
Datos de RS	MODIS, SRTM otros	MODIS, SRTM otros	-	COSMO – Aster	TBD	TBD

Tabla 1: Resumen de los requerimientos principales del usuario.

dónde:

- planilla de estratificación: mediante esta, el usuario ingresa su análisis, datos que serán usados posteriormente en el cálculo de los algoritmos.
- ERDN: Estratificación Riesgo de Dengue Nacional.
- ERDU: Estratificación Riesgo de Dengue Urbano.
- ERDNAmb: Estratificación Riesgo de Dengue Nacional Ambiental.
- ERDNfact: Estratificación Riesgo de Dengue Nacional Factorial.
- ERDUAmb: Estratificación Riesgo de Dengue Urbano Ambiental.
- ERDUfact: Estratificación Riesgo de Dengue Urbano Factorial.
- TBD: To Be Defined, A ser definido por el Ministerio de Salud en un futuro próximo.

Para una descripción completa de los requerimientos del usuario ver [17].

2 Arquitectura

La Comisión Nacional de Actividades Espaciales desarrolla software bajo los estándares de la European Space Agency

(ESA) los que definen prácticas para el desarrollo de software espacial y deben ser aplicadas en este ámbito[3]. La especificación de requerimientos, el diseño detallado del sistema, y la definición de sus interfaces son etapas que deben ser llevadas a cabo en el desarrollo de un correcto sistema informático que se desarrolle en el ámbito espacial. De este modo, cualquier software desarrollado en CONAE, y en concreto el SAT/ERDNU, debe cumplir con dichos estándares. Además el sistema es diseñado para operar dentro de una arquitectura previamente establecida: el CUSS (atención al usuario en CONAE ver [6]) y cumpliendo con sus especificaciones. El propósito de esta sección es describir el diseño de la arquitectura del sistema SAT/ERDNU.

Un sistema será dividido en partes: subsistemas y unidades. Por unidad se entiende a la mínima porción del sistema que puede ser desarrollado independientemente. Mientras que un subsistema podrá agrupar a un conjunto de unidades con un objetivo en común.

Este sistema está compuesto por los subsistemas Data Translation (DT), Algorithm Executor (AE), Spatial Data (SD), Visualization (VZ) y el subsistema Alarm Trigger (ver figura 2).

Subsistema DT (Data Translation), es el responsable de cargar en el servidor los datos enviados por el usuario; validar los datos enviados por el usuario y convertir los datos a XML. Una vez convertidos a XML, los datos son enviados al subsistema SD, para luego ser consultados por el subsistema AE. Este subsistema contiene las unidades Uploader WebInterface, xls2xml, y las unidades Translator (una por tipo de planilla a guardar).

Subsistema SD (Spatial Data). El subsistema SD, será el encargado de almacenar tanto la salida del subsistema DT como la salida del subsistema AE de manera óptima para ser luego usados por el subsistema VZ.

Subsistema AE (Algorithm Executor). El subsistema AE, será el responsable de ejecutar los algoritmos para producir los mapas de riesgo (un algoritmo por mapa de riesgo), para el período de tiempo especificado en las reglas, y con los datos injertados por el subsistema DT.

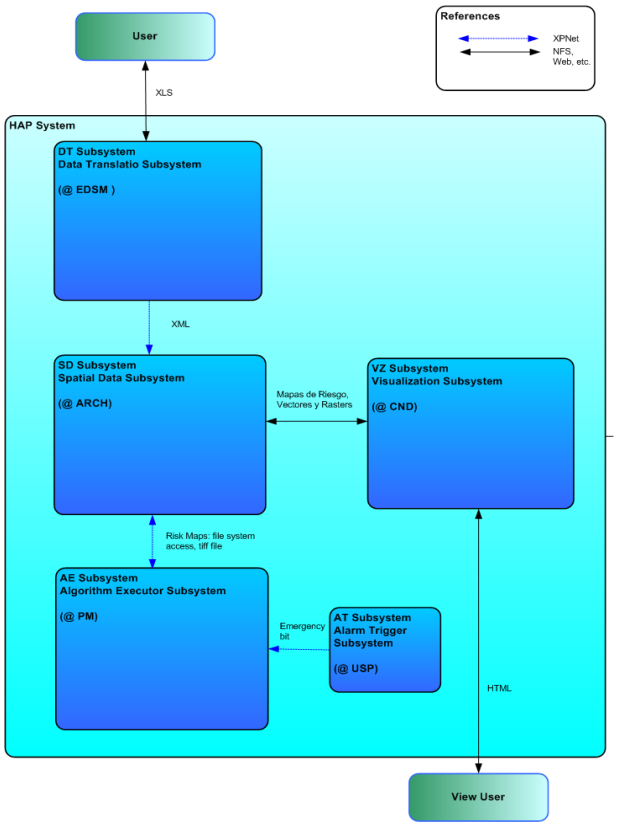

Figura 2: Representación arquitectural del sistema SAT/ERDNU.

Subsistema VZ (Visualization) será el encargado de mostrar los datos de manera amigable al usuario, a través de la web.

Este Subsistema implementará la descarga de los mapas de riesgo y de las capas de información disponibles a la hora de la visualización. Además mediante este Subsistema se permitirá la impresión de mapas.

Subsistema AT (Alarm Trigger) será el encargado de activar la situación de emergencia; en esta situación el AE cambiará la frecuencia de ejecución y los parámetros de los algoritmos. El manejo del pedido de emergencia es gestionado por las unidades Emergency Interface y el Emergency Module.

El presente diseño, como se mencionó anteriormente, además de ser ideado para su reusabilidad en otros SAT, fue pensado para ser instalado en un entorno productivo cumpliendo con la arquitectura general del CUSS de CONAE [14] Es así, que al enfrentarnos con esta aplicación informática fue necesario pensar primero en su operatividad, asegurando, de esta manera su correcta inserción en el ambiente productivo. Para un mayor detalle ver [15]

2.3 Subsistema Data Translation

Las unidades que conforman el subsistema DT de HAP son (figura 3):

Unidad Uploader WebInterface. Esta unidad es la encargada de brindar la interface gráfica al usuario para que el mismo suba los datos de planillas al servidor. Además esta unidad es la encargada de mostrar un mensaje de éxito o fracaso según como haya sido la carga de los datos. En el caso de fracaso, se describirá donde se produjo el error y brindará esta información al usuario.

Unidad xls2xml. Esta unidad es la encargada de convertir el archivo xls subido por el usuario autorizado a archivo xml. Es así que la función principal de dicha unidad es convertir los datos ingresados a un formato estándar y de fácil acceso para su posterior procesamiento. Dicha unidad también guarda el archivo xls en una carpeta input como respaldo informático de la carga del sistema.

Unidades Translators. Existe un translator por cada tipo de planilla ingresada por el usuario. La función principal de estas unidades es la verificación de los datos cargados por el usuario y el almacenamiento en el subsistema SD de los mismos. En el caso en el cual los datos sean erróneos, esta unidad se encarga de disparar una respuesta al usuario.

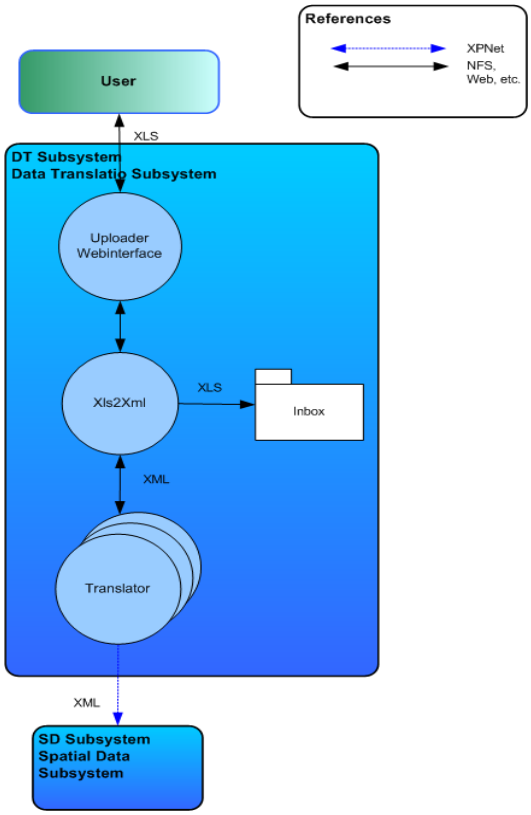

Figura 3: El subsistema DT

2.4 Subsistema Spatial Data

Este subsistema es el encargado, en primera instancia, de almacenar los archivos xml correspondientes a los datos ingresados por el usuario. Los cuales son necesarios para la ejecución de los algoritmos. En segunda instancia, almacena los resultados de la ejecución de los algoritmos en formato tiff para su posterior visualización. Es así que el subsistema SD está compuesto por una unidad que es la encargada de copiar los diferentes archivos en las carpetas de almacenamiento. Es requisito de la actual arquitectura del CUSS de CONAE, no disponer de bases de datos propietarias y tender de esta manera a manejar la mayor parte de los datos a través de archivos prescindiendo del uso de bases de datos. Por este motivo es que los datos tanto ingresados por el usuario como los producidos por la aplicación de los modelos se almacenan en archivos xml y tiff respectivamente. Dicha unidad maneja conflictos de accesos y atomicidad de operaciones. La figura 4 muestra dicho subsistema.

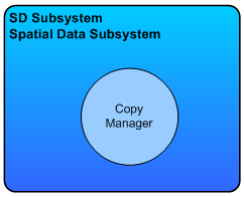

Figure 4: El subsistema SD

2.5 Subsistema Algorithm Executor

Las unidades que conforman dicho subsistema son (ver figura 5):

Algorithm Execution Manager. Esta unidad gestiona la ejecución de los algoritmos específicos para la producción de los mapas de riesgo. Las reglas son archivos XML donde se especifican los distintos parámetros necesarios para la ejecución correcta de los distintos algoritmos. Esta unidad guarda los mapas de riesgo producidos por cada algoritmo en el subsistema SD .

Process_1,Process_2,...,Process_n. Existe una unidad Process por algoritmo a ejecutar. El mismo es robusto, es decir, posee reglas de ejecución aun cuando no se cuente con todos los valores de las variables intervinientes. De esta manera, simplemente agregando una unidad Process, se podrá agregar nuevos productos al sistema.
xmlParser. Esta unidad será la encargada de la interface entre el archivo xml y los datos necesarios para la ejecución de cada algoritmo.

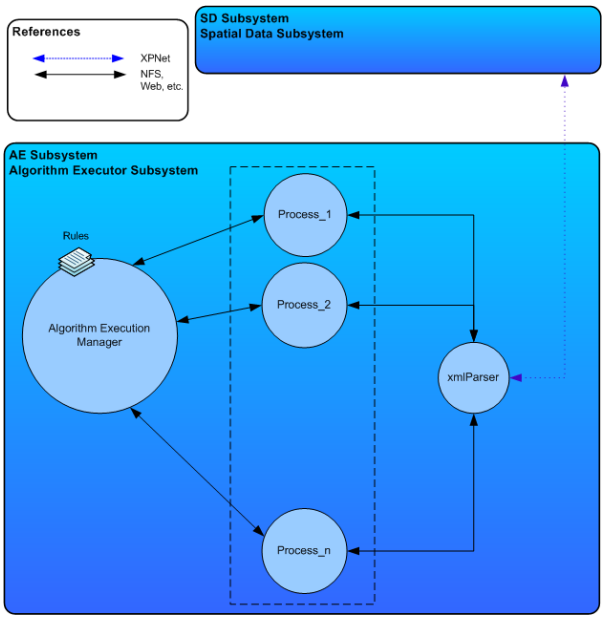

Figura 5: El subsistema AE

2.6 Subsistema Visualization

Las unidades que conforman dicho subsistema son (ver figura 6):

GeoServer. Esta unidad proporciona diferentes datos geográficos, entre ellos los mapas de riesgo generados por los algoritmos.

GIS Web Viewer. Esta unidad muestra al usuario autorizado los datos proporcionados por la unidad GeoServer, de manera amigable e interactiva. Dicha funcionalidad es lograda mediante un sistema de información geográfico (SIG). Además esta interface permite la descarga de las capas visualizadas (el usuario necesita tener permisos de descarga) y permite la impresión de mapas (aquí también el usuario deberá tener permisos para tal fin).

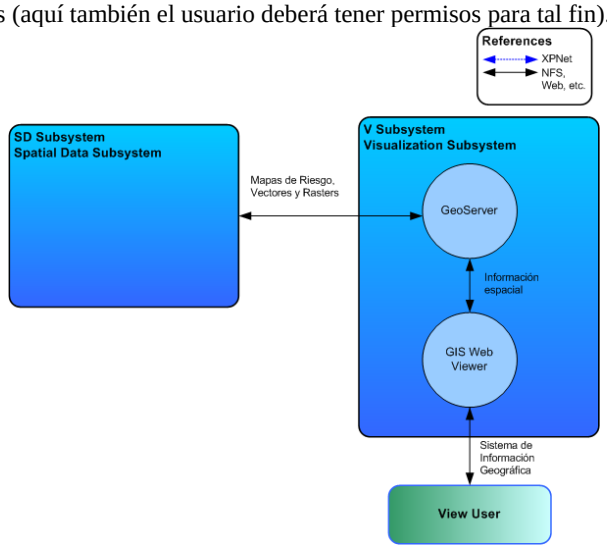

Figura 6: El subsistema VZ

2.7 Subsistema Alarm Trigger

Las unidades que componen el sub sistema AT son (ver figura 7):

Unidad Emergency Interface. Esta unidad será la encargada de brindar la interface para que el al usuario autorizado active la situación de emergencia.

Unidad Emergency Manager. Esta unidad será la encargada de modificar las reglas de los algoritmos en el subsistema AE, mediante las cuales se procesan los datos.

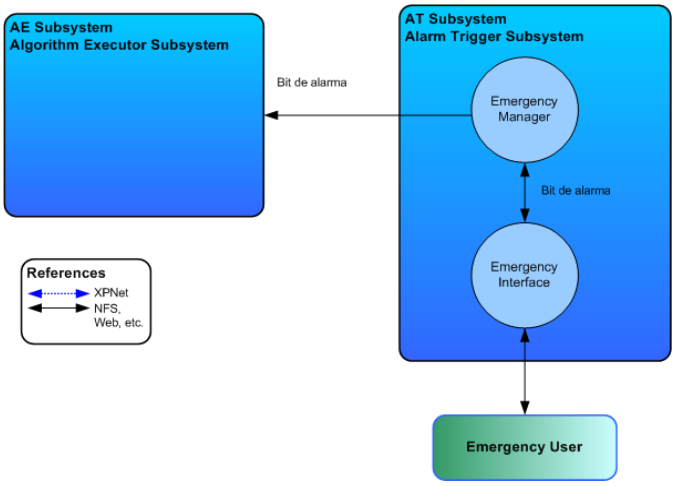

Figura 7: El subsistema AT

3 Implementación y tecnologías elegidas

Uno de los mayores beneficios del desarrollo orientado a objetos es la re-usabilidad, pero para alcanzar este objetivo es necesario un correcto diseño. Diseñar software orientado a objetos no es fácil ni trivial. Es necesario encontrar objetos adecuados, crear sus clases y definir jerarquías de herencia e interfaces, teniendo en cuenta sus interrelaciones. El diseño debería ser especifico del problema pero suficientemente general como para poder reutilizar la solución en problemas y requerimientos futuros, evitando re-diseño o por lo menos reduciéndolo [1]. El desarrollo de software orientado a objetos creció significativamente durante los últimos años. Esto es debido a un crecimiento en las técnicas de modelado, los Patrones de Diseño y finalmente al surgimiento de nuevos frameworks de trabajo [21].

Los Patrones de Diseño proveen una manera eficiente de crear software orientado a objetos [4] más flexible, elegante y re-utilizable. Por lo tanto su correcta aplicación en el diseño de los sistemas influirá significativamente en su reusabilidad, flexibilidad, extensibilidad y mantenibilidad. El crecimiento exponencial de las aplicaciones web ha dado lugar a un complejo conjunto de las mismas, donde el mantenimiento, la flexibilidad y la extensibilidad se ha vuelto extremadamente difícil [21], hecho que acentúa aún más la necesidad de una correcta aplicación de Patrones de Diseño. Los OSS pueden ser definidos como "Software donde el código fuente está disponible para usar, estudiar, reusar, modificar, mejorar y redistribuir por los usuarios del mismo [12]. Actualmente, instituciones y compañías tienen que enfrentarse al desarrollo de proyectos ampliamente distribuidos con requerimientos que cambian permanentemente. En [24] se plantea la adopción de prácticas OSS para mejorar el desarrollo de este tipo de proyectos y muestra como las pueden ser aplicadas.

A partir de estas premisas y teniendo en cuenta la necesidad de incrementar la productividad disminuyendo complejidad se decidió integrar un conjunto de tecnologías Open Source disponibles a fin de facilitar la creación de SAT/ERDNU. En general cada una de las tecnologías elegidas presenta un Patrón de Diseño acorde a los resultados que se quieren obtener. Es decir que la elección de las herramientas tecnológicas no es un tema menor en la construcción de todo el SAT/ERDNU. Además de todo lo mencionado un correcto Patrón de Diseño asegura una mayor facilidad de mantenimiento de nuestro software.

3.1 GeoServer

GeoServer es un servidor de información espacial Open Source escrito en Java que permite a los usuarios compartir y editar datos espaciales. Este publica datos usando estándares abiertos (open standards). GeoServer es la implementación de referencia del Open Geospatial Consortium (OGC) [5] y tiene implementado los protocolos de Web Feature Service (WFS), Web Coverage Service (WCS) y Web Map Service (WMS). Es uno de los servidores de mapas más estables en cuanto a tecnologías open source se trata y fue usado en el subsistema VZ, unidad GeoServer para almacenar y manejar la publicación de los datos geográficos a través de Internet. GeoServer es la unidad de administración de los datos geográficos, mediante el cual se accede de forma correcta a las capas de información a ser visualizadas por el usuario. En una vista de alto nivel, GeoServer, consiste de muchos módulos que interactúan entre sí. Todos estos módulos están conectados usando Spring IOC (Inversión of control[1]), de esta manera un módulo en tiempo de ejecución puede hacer uso de servicios provisto por otras clases. La figura 8 muestra esta arquitectura de alto nivel.

3.2 Geomajas

Geomajas es un complejo framework de visualización GIS Open Source y es usado para lograr la correcta visualización de los datos brindados por el Servidor de Mapas. Presenta una arquitectura cliente-servidor que facilita la visualización y edición de datos geográficos en la web. Geomajas tiene seguridad integrada y es escalable. Es compatible con los estándares OGC tales como WMS y WFS soportando además conexión directa a bases de datos espaciales [8]. Además es programado en Java sobre Spring [22] lo cual nos asegura un diseño y una implementación correcta desde el punto de vista de diseño. Spring es una plataforma de desarrollo por capas para aplicaciones Java/J2EE, basado en el código publicado en [2]. Un GIS que sirve sus datos a través de Internet es intrínsecamente complejo en lo que se refiere a performance y programación. Dicha complejidad se desprende principalmente de la gran cantidad de tipos de conexiones disponibles, debido a esto es necesario tener particular cuidado de la arquitectura y diseño a adoptar. El uso de Geomajas y de Spring en la unidad de GIS Web Viewer, permite despreocuparnos de detalles bien resueltos en este tipo de frameworks, dejando la responsabilidad del diseño de las unidades a Geomajas y con él, a Spring. Así, Geomajas a través de Spring implementa un patrón de diseño bien conocido: el Model View Controler (MVC) [1]. MVC consiste en tres tipos de objetos: el modelo (Model) es el objeto aplicación, la vista (View) es su presentación, mientras que el controlador (Controller) define la manera en que la interface de usuario reacciona a las entradas del usuario. MVC desacopla estos objetos para incrementar flexibilidad y re-uso. Como ventaja principal se puede mencionar que dicho

1 El IoC es un patrón de diseño y un conjunto de técnicas de programación en donde el flujo de control de un sistema es invertido en comparación con el tradicional modo interactivo. En IoC es el framework quien llama a los componentes de la aplicación y no la aplicación la que llama al framework como en la programación tradicional.

patrón aísla la lógica de negocio de las consideraciones de la interface de usuario, de esta manera es fácil modificar o bien la apariencia visual o bien las reglas de negocio. El View fue implementado usando el GWT face de Geomajas. Google Web Toolkit (GWT) permite crear aplicaciones AJAX en Java, separando el front-end del back-end y permitiendo que la programación Java en el front-end sea luego compilada a JavaScript ejecutable [9]. En la figura 9 se puede observar la arquitectura de Geomajas [8].

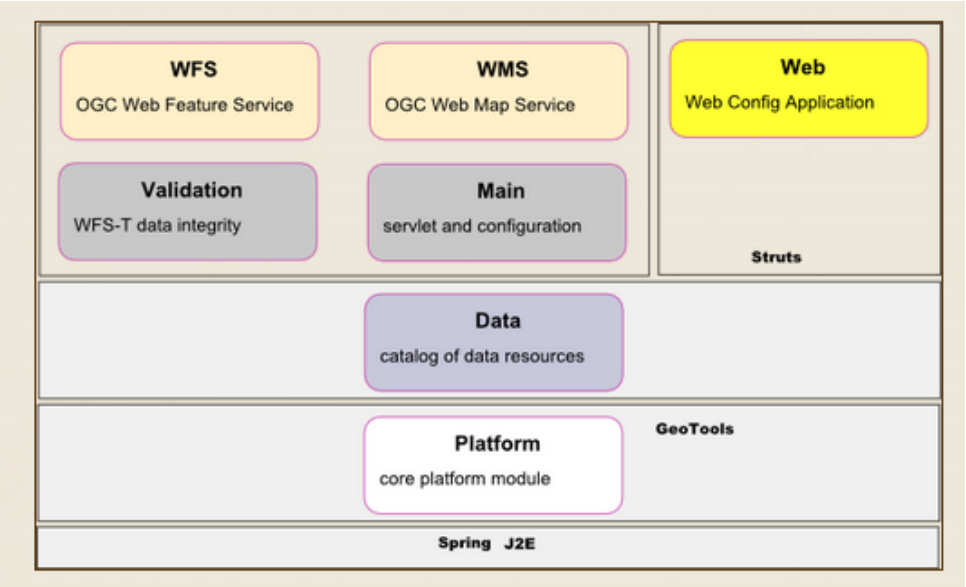

Figura 8: Arquitectura modular de GeoServer

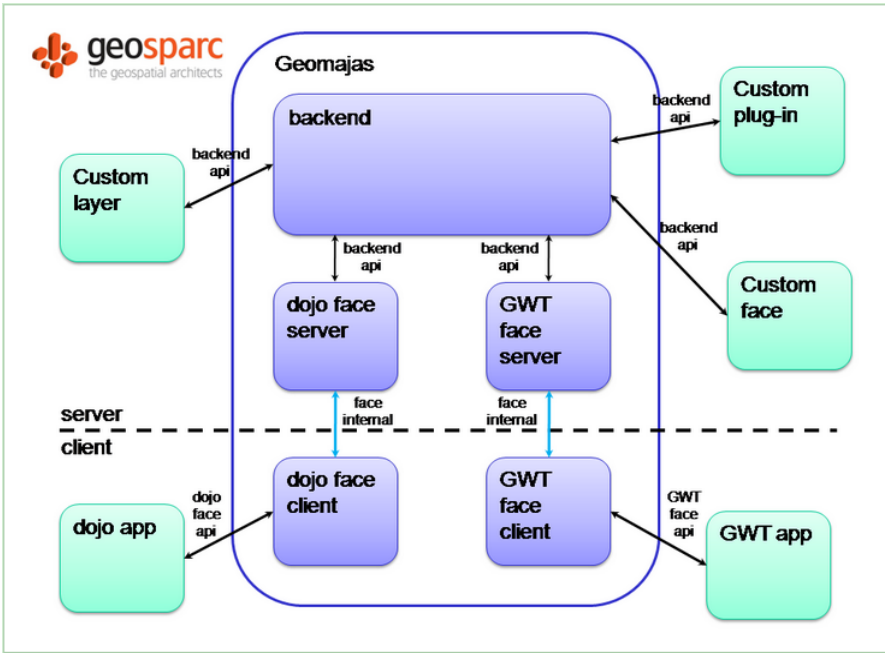

Figura 9: Arquitectura de Geomajas, extraido de "Geomajas user guide for developers" capitulo Architecture

Otro punto a favor de la elección de Geomajas para la implementación de la unidad GIS Web Viewer es su soporte para la autenticación de usuarios y manejo de roles y permisos, pues ese es un requerimiento del usuario.

3.3 Interface GeoServer - Geomajas

La comunicación o interface entre estas 2 unidades pertenecientes al subsistema VZ se realiza mediante el protocolo

WMS y el protocolo WFS. De esta manera se garantiza modularidad y flexibilidad. El hecho de tener los datos geográficos en un servidor de mapas y solo visualizarlos con Geomajas permite la centralización de los mismos, aspecto critico si tenemos en cuenta que los datos manejados por los SATs son intrínsecamente sensibles.

3.4 Spring Web Service

Como se vio anteriormente el subsistema DT es el encargado de recibir e ingresar las planillas con datos producto del relevamiento y evaluación del usuario y del manejo de los mensajes al usuario. En particular en el SAT/ERDNU es necesario validar un conjunto de datos pertenecientes a diversas fuentes. Realizar las Validaciones en la Planilla ingresada por el usuario, guardar dicha planilla en formato xml para ser leída por el subsistema SD, enviar un correo de verificación al encargado de los datos acerca de la aceptación o no de dicha planilla y comunicar el resultado de dicha validación al usuario son algunas de las tareas que en particular se realizan para satisfacer los requerimientos a nivel Nacional (ERDN). Toda esta funcionalidad fue implementada mediante un Web Service que realiza dichas tareas. La unidad Uploader es la encargada de subir el archivo xls al servidor. Luego, un analizador de xls (unidad xls2xml) traduce la planilla a formato xml a ser enviado por un Web Service consumer al Web Service provider. Una nueva planilla en el sistema es fácilmente agregada, mediante la creación de un nuevo Web Sevice. Un cambio en alguna planilla existente, es fácilmente incorporado modificando el contrato entre en consumer y el provider de dicho Web Service. De esta manera se garantiza versatilidad y re-uso de cada unidad en el subsistema o al menos, de su arquitectura. Incluso el proceso de validación puede cambiar y aun así, la arquitectura encajará. Supongamos el caso en que es necesaria la validación de la planilla por parte de un encargado en el proceso. Sabemos que datos de salud son sensibles y pensar en que el sistema puede requerir que una persona valide los datos por planilla no es descabellado. En este caso la planilla se transformará, hipotéticamente, en una orden que será procesada y que necesitará tiempo para tal fin. En este caso la arquitectura será la misma, el Web Service será el mismo, solo será necesario implementar nuevamente el método validación que devuelva una respuesta no inmediata cuando el estado de la orden sea aceptada

3.5 Interface: Uploader - Web Service

En el proceso de ingreso de datos debemos considerar la posibilidad de una numerosa cantidad de mensajes de retorno, los cuales pueden ser a su vez de distintos tipos y más aún cuando se trata de una numerosa cantidad datos a validar e ingresar al sistema. Es por ello que la interface entre el Web Service Consumer y la unidad Uploader se implementa a través del uso de respuestas xml. De esta manera las respuestas por parte del Web Service Provider son enviadas al Web Service Consumer quien luego escribe la misma en formato xml y es leída por la unidad Uploader. Dicha respuesta está compuesta por un estado y una descripción. Dependiendo del error detectado y su origen, el Web Service Provider establece la descripción del error a través de Apache log4j, librería usada para establecer el log de los fallos en la entrada de datos.

3.6 Algoritmos

Los modelos matemáticos que luego se transforman en algoritmos ver [12] que ejecuta el subsistema Algorithm Executor, son actualmente desarrollados por profesionales de distintas disciplinas. Su origen multidisciplinario hace difícil la implementación de los algoritmos en un lenguaje de programación unificado. Para resolver este problema es necesario encapsular la implementación de los algoritmos brindando interfaces homogéneas a la unidad manager de algoritmos: wrappers. De esta manera se logra independencia de la plataforma, pues un wrapper brindará la interfase necesaria para que el Algorithm Manager pueda cumplir su tarea y se abstraerá de la tecnología usada en su implementación. Como se puede ver en la arquitectura los subsistemas y unidades son lo suficientemente modulares para su correcta implementación en cualquier entorno de producción logrando una gran versatilidad.

3.7 Consideraciones

En general, los proyectos OSS presentan una pobre documentación en su desarrollo, falencia que se agrava con la ausencia de comentarios dentro del código [25]. Es obvio que desarrollos en estas condiciones requieren de mayor esfuerzo para codificar nuevas funcionalidades. El presente trabajo no es un caso aparte siendo necesario un completo análisis de las herramientas OSS a utilizar. En Geomajas, por ejemplo, existen manuales en su sitio web [8] que aún no están terminados y que son importantes a la hora de su configuración, desarrollo y mantenimiento. Otro inconveniente técnico abordado es la dificultad en converger un entorno estable de desarrollo. La gran cantidad de tecnologías que intervienen, deben ser entrelazadas armónicamente para que puedan interactuar correctamente, pero mientras mayor es el número de las mismas, mayor es la complejidad para lograr su correcta operatividad. El testing es otro de los puntos débiles que se encuentran en los OSS. Los mismos son frecuentemente entregados al público con un mínimo testing. En [25] se establece como justificación a este punto el hecho de que en OSS, el contribuidor (desarrollador que colabora en el desarrollo) tiene la libertad de trabajar en cualquier componente del proyecto. Ningún trabajo es asignado y ningún tiempo límite es asignado. Por lo tanto, en general, el participante tiende a estar interesado en tareas como el desarrollo, dejando de lado al testing.

4 Funcionalidad del SAT/ERDNU

En la tabla 2 se puede observar como son satisfechos los requerimientos del usuario (QUE) por parte de la infraestructura informática implementada (COMO). Así, mediante el subsistema DT (sistema de ingreso de datos) se logra la automatización de la carga de datos, el control de fallos en la carga de los mismos y en menor medida proporciona flexibilidad en la ejecución de los algoritmos ya que mediante un cambio en la planilla, un algoritmo en particular puede ser modificado sin necesidad de una cambio en su implementación. El subsistema Algorithm Executor y las unidades que lo conforman definen el módulo de procesamiento de datos. Mediante el mismo se satisface la necesidad de integrar la información ambiental al modelo beneficiando la correcta estratificación del riesgo. El modelo es alimentado tanto por la planilla de estratificación enviada por el usuario como de información ambiental que es proporcionada por distintos sensores remotos. De la misma manera el establecimiento de vectores como base del sistema de visualización produce un sistema integrado a multiescala que permite el análisis socio-económico a nivel tanto nacional como urbano y su relación con el riesgo de dengue presente. La encapsulación de los algoritmos usando wrappers permite una mayor flexibilidad en sus inputs, lo que deviene en que la incorporación de inputs de cualquier tipo a los algoritmos sea posible. En particular, incorporar imágenes SAR es factible y favorece la flexibilidad de los algoritmos (imágenes SAR muestran información tanto de día como de noche, en condiciones de nubes o sin ellas.) Como es notorio, el servidor de mapas puede ser accedido a través de WMS o WFS independientemente de la plataforma Web GIS desarrollada. Esto es, el uso de herramientas GIS como GVSig, QGIS, GRASS es posible y de esta manera se consigue una mayor flexibilidad de uso. Además el hecho de acceder a los datos mediante la plataforma de visualización WEB desarrollada garantiza el acceso desde cualquier plataforma con conexión a Internet y un navegador Web.

Requerimientos del Sistema COMO / Requerimientos del Usuario QUE?	Sistema de ingreso de datos	Modulo de procesamiento de datos	Base vectorial de datos	Imagenes Opticas LRes*	Imagenes Opticas HRes	Plataforma SIG (coordenadas geograficas)	SAR ImAges	Archivo de datos, metadatos y sub-productos (CUSS)	Basado en plataforma WEB	Acuerdo inter-institucional
Sistema Integrado Multiescala			9	9	9	9	3	3		
Automatizado /Usuarios y carga de datos	9		3					9	9	3
Control de fallos de carga datos y de coordenadas	9		9			9			3	
Incorporación datos ambientales, socio-económicos y vectoriales a nivel nacional/urbano		3	9	9	9	9	3	3		9
Visualización de riesgo a nivel nacional y urbano			3	3		9		3	9	
Cálculos e integración de información de amb y vectores		9	3		3	9		9	9	
Flexibilidad de algoritmos (activ. de estado de emergencia)	3	9			9		9	3	9	3

Tabla 2: lista de los requisitos de usuario principal y los requisitos del sistema que satisfacen los primeros. Proporciona una forma de analizar la posibilidad de dar una respuesta (o no) a cualquier exigencia del consumidor. Los números indican la posibilidad de respuesta.

El sistema facilita el ingreso de datos útiles a la hora del cálculo de los algoritmos y la ejecución de modelos mediante una interfaz de usuario amigable (figura 10). Mediante esta interfaz se informa al usuario acerca del éxito o fracaso en la carga. Además el sistema permite el manejo de usuarios con sus respectivos roles tanto para la carga de datos, visualización de los mismos, como así también su descarga. Con respecto a la visualización de los datos tanto de base, como los producidos por la ejecución de los modelos y algoritmos, pueden ser observados mediante la unidad Web GIS Viewer (figura 11). El usuario puede interactuar con los datos espaciales mediante herramientas base de cualquier GIS como lo son las herramientas de zoom, paneos y consultas geográficas (figura 13). Entre los datos de base con los cuales el usuario puede interactuar se presentan: localidades de la República Argentina, rutas y caminos de la República Argentina, ríos y cuerpos de agua de la República Argentina. Entre los datos producidos por los modelos encontramos el

vector y raster de la estratificación del Riesgo del Dengue a Nivel Nacional, el vector y raster representando el riesgo urbano, así como datos de imágenes satelitales (por ejemplo serie de Temporal de temperatura de MODIS) utilizadas en el cálculo y el procesamiento de los modelos. Internamente el sistema cada el periodo de tiempo establecido en un archivo de configuraciones ejecuta algoritmos tanto a nivel nacional como urbano para producir los mapas de riesgo de dengue estratificados. Esta funcionalidad es lograda mediante procesos batch desarrollados en Java y que son transparentes al usuario, es decir que el usuario del sistema solo interactúa con la visualización de los datos. Mientras que internamente el SAT/ERDNU maneja la ejecución de los modelos desarrollados (ver [12])

Figure 10: Sitio Receptor de Planillas: unidad Uploader

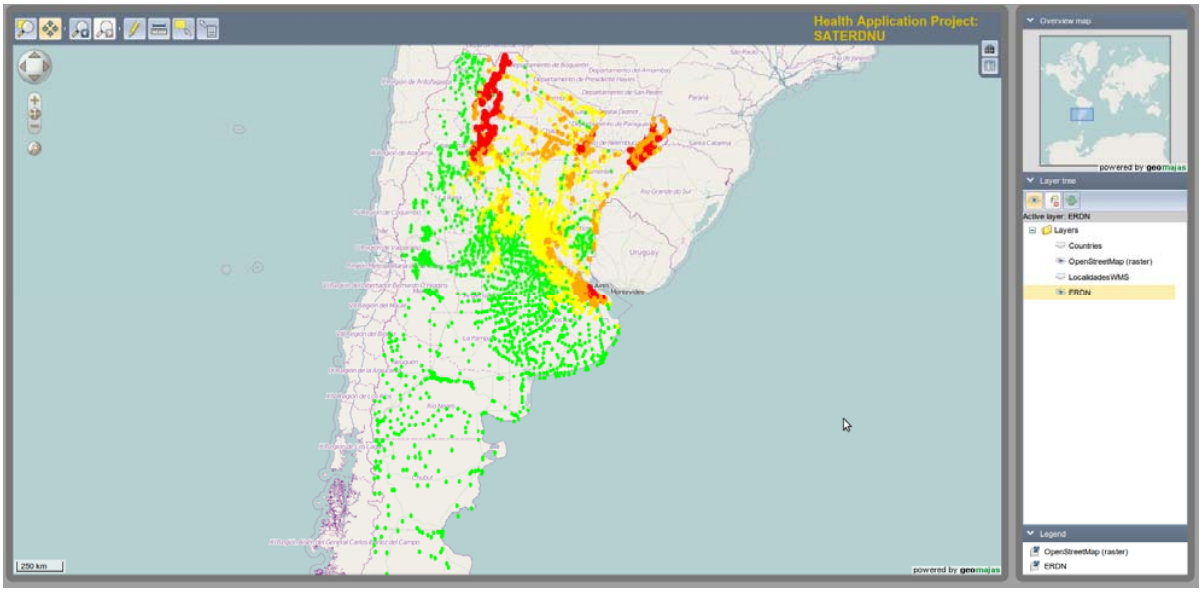

Figure 11: Sitio Visualizador: unidad Web GIS Viewer

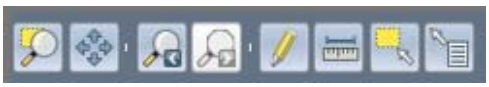

Figure 12: Herramientas del Visualizador Web

5 Conclusión

El presente trabajo describió de manera completa el diseño y la implementación del sistema SAT/ERDNU. El SAT/ERDNU brinda soporte a la prevención y el control estratégico del vector del dengue en Argentina y puede ser reutilizado en cualquier SAT. Integrando herramientas Open Source como Geomajas, GeoServer, Maven y Eclipse, se

garantiza un desarrollo adecuado en tiempo y calidad, de bajo coste y favoreciendo aún más la re-usabilidad (el uso de OSS elimina las posibles restricciones de distribución que pudieran tener herramientas propietarias). El uso del frameworks ampliamente aceptados por la comunidad como Spring facilita el mantenimiento de la aplicación, el MVC y la inversión del control dan un diseño conocido a las unidades intervinientes, favoreciendo las posibles modificaciones (obviamente es más fácil modificar sabiendo donde es necesario hacerlo). El diseño es altamente modular permitiendo además de reutilización en diferentes SATs, flexibilidad de operación (el sistema puede ser instalado correctamente y sin modificaciones en cualquier ambiente operativo). En general, proyectos Open Source presentan dificultades a tener en cuenta: pobre documentación que dificulta el desarrollo, necesidad de integración de numerosas tecnologías que dificultan la convergencia de todas ellas, y la no certeza en el reconocimiento de bugs (pobre documentación conjugada con pobre testing hace a veces pensar que un bug no es tal, que es un problema o bien de integración o bien de comprensión de la herramienta). Sin embargo, como lo demuestra el presente trabajo, es posible desarrollar una compleja aplicación operativa re-utilizable, flexible y mantenible usando herramientas Open Source de alta calidad respondiendo a estándares de primer nivel como los de la Agencia Espacial Europea en respuesta a los requerimientos de organismos de salud.

Referencias

[1] Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley, 1994.

[2] Expert One-on-One J2EE Design and Development. Wrox, 2002.

[3] EUROPEAN SPACE Agency,Bssc guides and reports. http://www.esa.int/TEC/Software_engineering_and_standardisation/TECT5CU% XBQE_2.html, MAY 2011.

[4] Booch, G., Rober, A. M., michael, W. E., and et al. Object-oriented analysis and design with applications. *Peking: Post and Telecom Press 3rd ed* (2008).

[5] OPEN GEOSPATIAL Consortium. http://www.opengeospatial.org/, May 2011.

[6] de La Toree, C. CUSS for SAC-D Architecture Design Document. Tech. rep., Instituto Gulich, CONAE, 2010.

[7] Domínguez, C., and Lagos, S. Presencia de aedes aegypti (diptera: Culicidae) en la provincia de mendoza, argentina. *Rev Soc Entomol Argent 60* (2001).

[8] Geomajas. Geomajas. http://www.geomajas.org/overview/about-geomajas, May 2011.

[9] Google. Google web toolkit. http://code.google.com/intl/es/webtoolkit/doc/1.6/overview.html, May 2011.

[10] Hales, S., de Wet, N., Maindonald, J., and Woodward, A. Potential effect of population and climate changes on global distribution of dengue fever: an empirical model. *Lancet* (2002).

[11] Kennedy, M. Evaluating open source software. *Defence AT&L* (2010).

[12] Lanfri, S., Frutos, N., Porcasi, X., Rotela, C., abd Estefania De Elia, G. P., Lanfri, M., and Scavuzzo, M. Algoritmo para la alerta temprana de dengue en un ambiente geomático. *Congreso Argentino de Informática y Salud 2011* (2011).

[13] Morasca, S., Taibi, D., and Tosi, D. Towards certifying the testing process of open source software: new challenges or old methologies? *Proceeding FLOSS. Proceedings of the 2009 ICSE Workshop on Emerging Trends in FreeLibreOpen Source Software Research and Development* (2011).

[14] Oglietti, M. Domain independent planning for space: Building a bridge from both shores. *International Workshop on Planning and Scheduling for Space* (2002).

[15] Peralta, G., and Elia, E. D. SAT/ERDNU Architecture Design Document. Tech. rep., Instituto Gulich, CONAE, 2011.

[16] Porcasi, X. *El Porceso de re-infestacion por T. infestans evaluado por sensores remotos*. PhD thesis, UNC, 2009.

[17] Porcasi, X. Requeriment baseline document. Tech. rep., Instituto Gulich, CONAE, 2011.

[18] Rossi, G. C., Lestani, E. A., and D'Oria, J. M. Nuevos registros y distribución de mosquitos de la argentina (diptera: Culicidae). *Rev Soc Entomol Argent 65* (2006).

[19] Scavuzzo, C. M., Lamfri, M. A., Rotela, C., Porcasi, X., and Estallo, E. Satellite image applied to epidemiology, the experience of the gulich institute in argentina. *E-Health. The International Trade Event and Conference fo eHealth, Telemedicine and Health ICT* (2006).

[20] Schneider, and Droll, D. A timeline for dengue in the americas to december 31, 2000, and noted first occurences. *Pan American Health Organization. Division of Disease Prevention and Control.* (2001).

[21] Shuang, L., and Chen, P. Developing Java EE Applicatrions based on utilizing design patterns. *WASE International Conference on Information Engineering* (2009).

[22] SpringSource. Springsource. http://www.springsource.org/, May 2011.

[23] Tauil, P. Urbanization and dengue ecology. *Cad Saude Publica* (2001).

[24] Torkar, R., Minoves, P., and Garrigós, J. Adopting free/libre/open source software practices, techniques and methods for industrial use. *Journal of the Association for Information Systems JAIS* (2011).

[25] Ullah, K., and Khan, S. A. A review of issue analysis in open source software development. *Journal of Theoretical and Applied Technology* (2011).

[26] WHO. World health organization. dengue: guidelines for diagnosis, treatment, prevention and control. *Geneva*

Congress (2009).

Datos de Contacto

Gonzalo Peralta. Instituto de Altos Estudios Espaciales Mario Gulich, Comisión Nacional de Actividades Espaciales. Centro Espacial Teófilo Tabanera, Ruta C45 km 8, Falda del Carmen (5187) Córdoba, Argentina. gperalta@conae.gov.ar
De Elia Estefania Aylén. Instituto de Altos Estudios Espaciales Mario Gulich, Comisión Nacional de Actividades Espaciales. Centro Espacial Teófilo Tabanera, Ruta C45 km 8, Falda del Carmen (5187) Córdoba, Argentina. deelia@gmail.com
Mario Alberto Lanfri. Instituto de Altos Estudios Espaciales Mario Gulich, Comisión Nacional de Actividades Espaciales. Centro Espacial Teófilo Tabanera, Ruta C45 km 8, Falda del Carmen (5187) Córdoba, Argentina. lamfri@conae.gov.ar
Sofía Lanfri. Instituto de Altos Estudios Espaciales Mario Gulich, Comisión Nacional de Actividades Espaciales. Centro Espacial Teófilo Tabanera, Ruta C45 km 8, Falda del Carmen (5187) Córdoba, Argentina. slanfri@conae.gov.ar
Ximena Porcasi. Instituto de Altos Estudios Espaciales Mario Gulich, Comisión Nacional de Actividades Espaciales. Centro Espacial Teófilo Tabanera, Ruta C45 km 8, Falda del Carmen (5187) Córdoba, Argentina. ximena.porcasi@conae.gov.ar
Nicolás Frutos. Instituto de Altos Estudios Espaciales Mario Gulich, Comisión Nacional de Actividades Espaciales. Centro Espacial Teófilo Tabanera, Ruta C45 km 8, Falda del Carmen (5187) Córdoba, Argentina. nfrutos@gmail.com
Camilo Rotela. Instituto de Altos Estudios Espaciales Mario Gulich, Comisión Nacional de Actividades Espaciales. Centro Espacial Teófilo Tabanera, Ruta C45 km 8, Falda del Carmen (5187) Córdoba, Argentina. crotela@conae.gov.ar
Marcelo Scavuzzo. Instituto de Altos Estudios Espaciales Mario Gulich, Comisión Nacional de Actividades Espaciales. Centro Espacial Teófilo Tabanera, Ruta C45 km 8, Falda del Carmen (5187) Córdoba, Argentina. scavuzzo@conae.gov.ar