Universidad de las Ciencias Informáticas

Título: Biblioteca para la habilitación de Sensores Web en el Sistema Integral de Supervisión y Control de Procesos Industriales Guardián del Alba

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autores:

Jorge Luis Acosta Alonso

Leiser Fernández Gallo

Tutores:

Ing. Yunior Bauta Pentón

Ing. Yolier Galán Tassé

La Habana, junio de 2015 .

"Año 57 de la Revolución"

Declaración de Autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Jorge Luis Acosta Alonso

_____

Firma del Autor

Leiser Fernández Gallo

_____

Firma del Autor

Ing. Yunior Bauta Pentón

_____

Firma del Tutor

Ing. Yolier Galán Tassé

_____

Firma del Tutor

Datos de contacto

Ing. Yunior Bauta Pentón

Ingeniero en Ciencias Informáticas desde 2009. Profesor instructor y especialista del centro DATEC con experiencia en temas de bases de datos postgresql. Programador del SCADA con 3 años de experiencias.

e-mail:ypenton@uci.cu

Ing. Yolier Galán Tassé

Ingeniero en Ciencias Informáticas se ha desempeñado como profesor, desarrollador e investigador, en la Línea de Sistemas Embebidos. Ha estado vinculado al área productiva al desarrollo de Sistemas de Gestión Industrial y SCADA con 5 años de experiencia.

e-mail: ytasse@uci.cu

Agradecimientos

A toda mi familia, en especial mi papá, mi mamá y mi hermana por estar ahí estos últimos casi 24 años. A mis compañeros de estudio y residencia de estos últimos cinco de universidad, cuyos nombres no me atrevo a mencionar por temor a dejar alguno fuera de la lista. A mis nakamas: Yugui, Sanji, Berseker, El Primo y sus Cachorros, Petter, Neto, Nakamura, Varo, Yuyu, Tanuky, Antu y Mr.Maker por ayudarme a crecer como persona mientras yo intentaba hacer lo mismo por ellos. A Jlacosa, otro de mis nakamas y compañero de tesis. A mis tutores, Leo, mi oponente y Naivy por su ayuda con la investigación. En fin, a todos los que me tendieron una mano amiga cuando me hizo falta, este título no fuera posible sin ustedes.

Leiser

A mis padres y hermano por estar siempre a mi lado y brindarme su apoyo incondicional en todo momento. Por saber guiarme por el buen camino hasta ser la clave de mis resultados y porque cada consejo fue una enseñanza que me sirvió para salir adelante. A mi compañero de tesis que juntos hicimos este sueño realidad y agradecerle que hoy soy una persona capaz de enfrentarme a una vida profesional , porque me enseñó a que no se necesita de alguien para aprender, así como que tenemos que ser capaces de superarnos por nosotros mismos. A mis tutores por saber guiarnos por el camino de la ciencia como todo un profesional. Siempre muy atentos y cuya fuerza de voluntad y entusiasmo son casi incomparables. Por siempre confiar en nosotros. A mis amigos (del grupo 5502, los del 5504, Claudia Pio, Alexander Martínez, Tania Martínez, Antuanett) y en especial a Antuanett por ser mas que una amiga y por compartir unas de las etapas mas difíciles de la universidad. A ellos por pasar tantos momentos buenos y malos juntos. A todos aquellos que de una forma u otra me ayudaron y me apoyaron en cumplir mi sueño.

Jorge

Dedicatoria

A mi mamá, mi papá y mi hermana. Al resto de mi querida familia. A todos mis nakamas que me ayudaron a encontrar el verdadero tesoro de One Piece: su amistad.

Leiser

A mi madre por ser quien me dio la vida y ser la persona a través de la cual miraba el futuro. Por crearme un punto de partida y ser ella mi punto de llegada. Por compartir junto conmigo este sueno y por ser ella a quien le he dedicado cada uno de mis logros. A mi padrastro por ocupar el lugar que pensé que estaría vacío para siempre. Por ser la persona que me ayudo a llegar a cumplir mis objetivos y metas. Por ser la persona que sacrifico todo y lucho para que hoy fuera ingeniero. A mi hermano por estar siempre apoyándome y porque es la persona en quien siempre pienso y doy lo mejor de mi para servir de ejemplo. A mis primos Fernando y Miguel Jesús por ayudarme a pasar la universidad, por cargar con mi estrés de los problemas y con mis alegrías. Por estar la mayor parte del tiempo conmigo. A mi tía Conchy por ser una persona que si fuera necesario me daría lo que no tuviera y especialmente por ser ella mi segunda madre. A mi abuelo por ser mi conciencia, por ser la persona a la cual siempre trato de imitar, por ser la persona que confía en mi con los ojos cerrados. A mi tía Teresita por ser la persona que hozo el mayor cambio en mi personalidad. Por enseñarme a que la vida no tiene limites y siempre tenemos nuevos objetivos que cumplir. Por acogerme y quererme como un hijo.

Jorge

Resumen

Sensor Web es una tecnología que permite compartir los datos medidos por sensores de diversos proveedores, así como sus metadatos, en un lenguaje común; de modo que los mismos pueden ser consumidos y procesados de una forma transparente al sensor en cuestión.

La gran cantidad de sensores que puede manejar el sistema de supervisión, control y adquisición de datos Guardián del Alba pudieran ser usados en conjunto con otros de otros sistemas industriales si estos expusieran sus datos y metadatos de forma homogénea.

El presente trabajo describe el proceso de construcción de una biblioteca para desarrollar sensores web que permitan compartir los datos recolectados por el sistema SCADA-GALBA en redes de sensores web.

Palabras Clave: habilitación de sensores web, SCADA-GALBA, red de sensores web.

Índice

Índice de Figuras

Índice de Tablas

Introducción

En la actualidad, los avances en la ciencia y la tecnología, se han convertidos en los pilares fundamentales de la sociedad. Los procesos automáticos se han hecho tan comunes que, más que de necesarios, se les cataloga de imprescindibles para el avance de la sociedad. Cuba no queda exenta de este proceso y tampoco la Universidad de las Ciencias Informáticas. De ahí la creación de el Centro de Informática Industrial (CEDIN), dedicado al desarrollo de software para el sector industrial.

Uno de los principales proyectos del CEDIN es el desarrollo del Guardián del Alba, que no es más que un Sistema de Supervisión y Control de Procesos Industriales, lo que se conoce en el campo de la informática industrial como SCADA por sus siglas en inglés, de ahí que al producto se le conozca como SCADA-GALBA.

En la pirámide de automatización que constituyen los SCADA, los sensores forman parte de la base de los elementos primarios de control. Estos deben transmitir los datos que censan en el campo mediante protocolos industriales. Estos últimos son procesados posteriormente, almacenados en bases de datos históricas y el estado del proceso que se esté monitorizando es inmediatamente mostrado al operador. No aprovechar esta cantidad de datos adquirida sería irracional; de modo que en el mercado mundial los sistemas de este tipo no se usan solamente para supervisar procesos, sino que son integrables con otros sistemas dedicados a control de estadísticas, análisis de información, toma de decisiones, predicciones o cualquier otra utilidad que puedas ser obtenida partiendo de los datos recolectados por un SCADA.

Las tecnologías avanzan aceleradamente y lo que hoy se considera novedoso y muy usado es propenso a ser reemplazado por algo nuevo dentro de pocos años; si un sistema no es lo suficientemente adaptable a las nuevas condiciones que aparezcan pasará a ser obsoleto y perderá competitividad en el mercado.

La habilitación de sensores web (en lo adelante SWE por las siglas en inglés de Sensor

Web Enablement) [1] es una tecnología creada por el Consorcio Geoespacial Abierto (conocido como OGC por las siglas en inglés de *Open Geospatial Consortium*) con el fin de compartir datos y metadatos de sensores de diversos proveedores mediante un lenguaje común. Estos sensores manejan la información que comparten de forma estandarizada, de forma tal que es posible la lectura de la información que brindan estos por cualquier aplicación, sin necesidad de que la misma conozca información particular de cada sensor. Existen varias organizaciones que emplean esta tecnología, entre ellas pueden ser citadas: La Red de Recursos Hídricos Australianos (*Australian Water Resources Network*), La Red de Información y Observación del Medio Ambiente Europeo (*European Environment Information and Observation Network*), y la Red de Observación de la Tierra Sudafricana (*South African Earth Observation Network*) [2].

Uno de los elementos fundamentales en SWE son los servicios de observación de sensores (en lo adelante SOS, por las siglas en inglés de *Sensor Observation Service*). Estos son 52°North servidores que se encargan de centralizar y exponer toda la información recogida por varios sensores web.

Para la comunicación con sistemas externos el SCADA-GALBA emplea dos vías, las cuales se pueden agrupar bajo el nombre de "subsistema Integración con terceros": un servidor que implementa la especificación OPC DA y el servidor Comm3Server el cual brinda Servicios Web. A través del servidor OPC DA los terceros pueden leer las variables que hayan sido configuradas y procesadas en el SCADA-GALBA pero solo aquellos que se encuentren sobre la plataforma Windows ya que la capa de comunicación OPC DA es dependiente de la tecnología COM/DCOM de Microsoft. Los servicios Web, por su parte, permiten que los terceros puedan acceder a servicios brindados, tales como: el intercambio de variables, alarmas, eventos y autenticación para el control de acceso a la información. Ambos servicios están dedicados a permitir a otras aplicaciones conocer información del SCADA-GALBA; sin embargo no exponen esta información como sensores web por lo que este no puede ser integrado con redes de sensores, de modo que la información que es capaz de brindar no es

accesible desde aplicaciones implementadas por terceros que usan esta fuente de información. En otras palabras, el SCADA-GALBA no puede intercambiar información con sistemas que reconozcan a los servidores de sensores web como fuente de información.

Diversificar las formas de integración será importante y atractivo para los clientes del SCADA-GALBA ya que mientras más facilidades y funcionalidades brinde en este sentido, el sistema será más integrable, y se podrá sacar mayor provecho de los datos adquiridos. Teniendo en cuenta que en el campo de la informática el término interoperabilidad se define como "la habilidad de dos o más sistemas, redes de comunicación, aplicaciones o componentes para intercambiar información entre ellos y para usar la información que ha sido intercambiada" [3] se plantea el siguiente:

Problema de la investigación: ¿Cómo lograr mayor interoperabilidad del SCADA-GALBA con otros sistemas en internet que permita el monitoreo de sus recursos en la web?

Por lo que el **objeto de estudio** es el proceso de gestión de la información para redes de sensores en internet, teniendo como **campo de acción** la interoperabilidad para redes de sensores en entornos industriales.

Objetivo general: Desarrollar una biblioteca que permita la habilitación de Sensores Web en el SCADA-GALBA para que pueda ser integrado con otros sistemas de internet.

Para cumplir con el mismo se proponen como **tareas investigativas**:

- Realizar una revisión bibliográfica sobre las tendencias actuales y los casos de éxitos relacionados con el tipo de sistema que se pretende desarrollar.
- Elaborar el marco teórico a partir del estado del arte actual del tema.
- Seleccionar tecnologías y herramientas para elaborar la solución del problema.
- Diseñar la solución propuesta.
- Implementar la solución propuesta.
- Validar la solución propuesta.

Durante el desarrollo de esta investigación se utilizan varios **métodos científicos que se describen a continuación**:

Empíricos:

- **Observación:** Para reunir información sobre el comportamiento de la interoperabilidad del SCADA-GALBA con otros sistemas de internet.
- **Pruebas:** Para validar el correcto funcionamiento de la solución desarrollada.

Teóricos:

- **Analítico-Sintético:** Con el uso de este método de investigación se analiza la información obtenida, mediante su desglose, para conformar el marco teórico de la investigación.
- **Histórico-Lógico:** Mediante este método se analizará la trayectoria y la evolución de las diferentes formas de interoperabilidad del SCADA-GALBA con otros sistemas de internet, así como la selección de las herramientas para el desarrollo de la solución propuesta.

Capítulo 1: Fundamentación Teórica

En este capítulo se abordan temas relativos a la interoperabilidad y accesibilidad de redes de sensores en entornos industriales. Definiciones de los tipos de sensores, los estándares que se implementarán en la solución del trabajo y los artefactos generados de la metodología de desarrollo.

1.1 Sensores

Un sensor es un dispositivo que convierte un parámetro físico, químico o biológico en una señal eléctrica, ya sea analógica o digital [4]. De entre los ejemplos más comunes pueden citarse los sensores para medir la temperatura (termómetro), la velocidad del viento (anemómetro), la conductividad o la radiación solar.

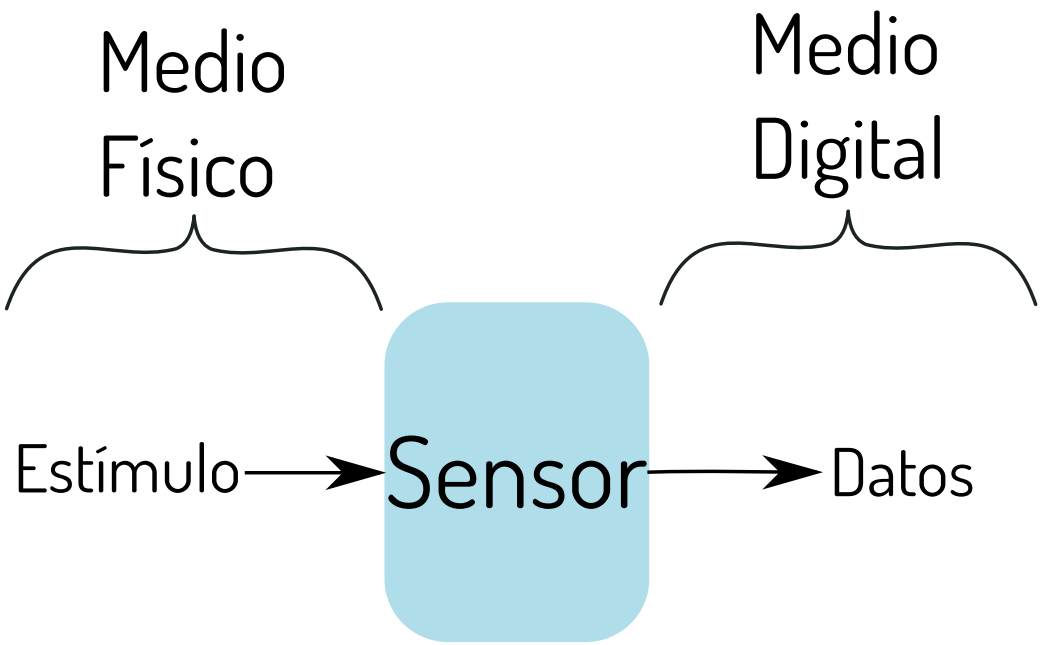
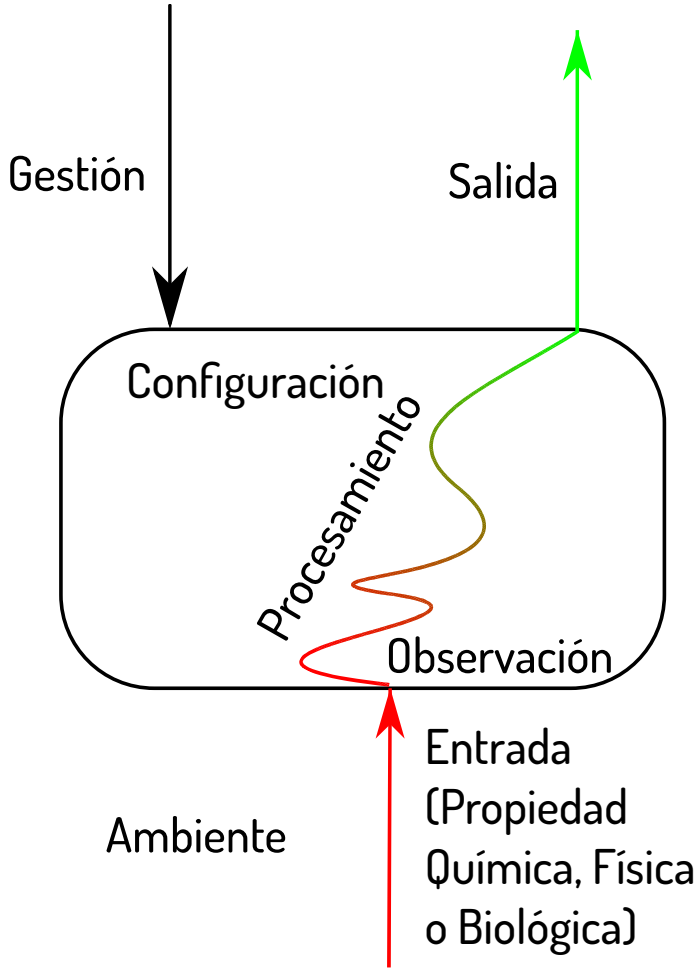


Figura 1: Sensor.

Las siguientes secciones proveen una mayor y detallada explicación de los sensores y la diferencia entre sensor simple, complejo y sistema de sensores.

1.1.1 Sensor de forma simple

Un sensor observa una propiedad medioambiental que pudiera ser biológica, química o física en el medio donde este se encuentra, en una posición y un instante de tiempo específicos, esto es: con un contexto temporal y espacial. La posición del sensor no tiene por qué coincidir con la del medio observado por él [5].



La propiedad observada suele ser transformada internamente en otro tipo de representación, ya sea eléctrica o mecánica por el propio sensor. A esa representación interna se conoce como señal. Dentro del sensor pueden ocurrir muchos tipos de procesos sobre la señal, como pueden ser la linealización[1], cálculo basado en la calibración de coeficientes,

1 Llevar algo a una forma lineal.

conversión a diferentes tipos de representación y cualquier otro cálculo para preparar la salida de los datos.

Aún cuando el camino que recorre la señal desde la observación hasta la salida toma cierto tiempo, el contexto temporal (el instante de tiempo en que se realizó la observación) y el espacial (el lugar donde se tomó), no pueden ser desechados. Así, en determinados contextos, el instante de tiempo en que se hace la observación y la localización de la misma pueden ser una información esencial.

Finalmente, la propiedad observada es accesible desde fuera del sensor en una representación procesable por computadoras. Diferentes sensores pueden representar diferentes representaciones de la misma propiedad observada. Estas pueden diferenciarse en el método de representación, el tipo de procesamiento interno que ha recibido la señal, la calidad de la representación o incluso la unidad de medición empleada. Las estimaciones del valor observado pueden ser un valor simple, un rango de valores, una selección de mejores y peores, una secuencia de valores o un arreglo multidimensional. Esto puede tener datos por punto espacio-temporal o bien una representación estadística en el espacio y el tiempo.

1.1.2 Sensor complejo

Si una propiedad no puede ser observada con un sensor en su forma simple, entonces es posible crear un sensor compuesto usando varios sensores simples. Este modelo compuesto se ilustra a continuación [6].

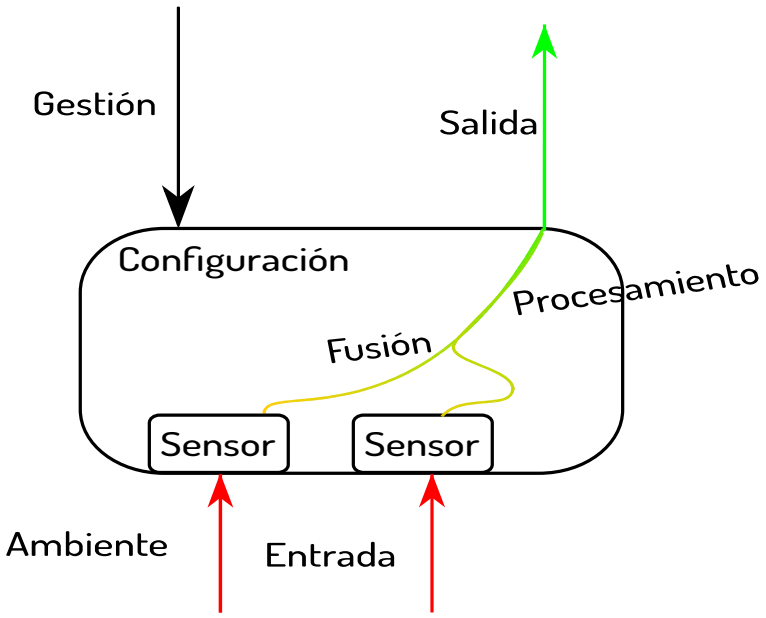


Figura 2: Modelo de un sensor complejo

La información sobre las propiedades observadas por los componentes individuales pueden ser procesadas por cualquier método de procesado de información, por ejemplo: fusión de bloques. La salida de los sensores de forma compleja está definida por las operaciones del sensor. Esto significa que la diferencia entre la salida de un sensor complejo y uno simple es transparente; incluso los complejos tienen que proveer información espacio-temporal a sus datos. De ahí que el contexto temporal de sensores de forma compleja sea una función temporal de los contextos de las observaciones individuales.

En dependencia del contexto de la aplicación, los sensores complejos pueden a su vez ser agregados en otros sensores complejos, pues la estructura interna de un sensor es una caja negra para el resto del sistema.

1.1.3 Sistema de sensores

Varios sensores pueden ser combinados dentro de un sistema de sensores que permita el

control de estos además de cada uno de ellos por separado, esto mediante una interfaz de control. Estos sistemas se caracterizan por tener una salida e interfaz de control que refleja su unidad organizacional, la que puede variar en tipo y naturaleza. Las partes individuales pueden proveer interfaces, aún cuando formen parte de un sistema de sensores.

1.1.4 Sensor Web

Sensores Web es un concepto empleado para lograr sistemas de recolección, fusión y distribución de datos de sensores; de forma colaborativa, coherente y consistente. Estos pueden ser vistos como un nuevo tipo de internet para monitorizar fenómenos espacio-temporales que ocurren en el ambiente físico. Cualquier tipo de sensor, desde un simple termómetro en una ubicación fija, hasta un sistema de sensores empotrado en un satélite que orbita la tierra; será hecho accesible a nivel global en un futuro no lejano.

Cada sensor después de ser desplegado es asociado al fenómeno que detectará, así como con el lugar donde este reside, esta información es luego almacenada en el sensor para el acceso posterior o directamente enviada a un sistema de acumulación. La recuperación y procesado de los datos del sensor, así como el control de los dispositivos, serán expuestos por medio de entidades que colaboran vía internet.

Los Sensores Web representan una metaplataforma que integra sensores arbitrarios y redes de sensores; cada uno mantenido y operado por instituciones individuales. Gracias a que el diseño arquitectónico de estos permite tanto la integración de sensores independientes como sistemas de sensores sin necesidad de grandes cambios en los sistemas legados.

1.2 La habilitación de sensores web

La habilitación de sensores web, mejor conocida como *Sensor Web Enablement* (SWE por sus siglas en inglés), iniciativa de OGC define estándares para la construcción de sensores web, en lo sucesivo SW. SWE permite la elaboración de sensores disponibles en la Web a través de formatos bien definidos e interfaces de servicios Web, ocultando la comunicación de sensores y los protocolos de sensores heterogéneos que trabajan en la parte superior de estos

sensores [7].

Actualmente, las redes de sensores son elaboradas alrededor de diversas comunidades de sensores y usuarios. De modo que cada una tiene su propio sistema para descubrir y acceder a observaciones, recibir alertas y encomendar tareas a sensores y modelos. Incluso cuando estas sean medianamente coherentes, cada tipo de sensor tiende a tener su propia semántica en los metadatos, sus propios formatos y su propio software. De modo que adicionar soporte para un nuevo sensor en uno de estos sistemas se traduce en días, meses e incluso años de trabajo acompañados de un alto costo.

De ahí que SWE haya sido diseñado para habilitar soluciones que satisfagan las siguientes necesidades:

1. **Descubrimiento de sensores, observaciones y procesos:** es deseable que sea fácil descubrir todos los datos que puedan brindar los sensores; lo cual es particularmente importante, por ejemplo: durante operaciones de rescate o mitigación frente a un desastre o ataque.
2. **Determinación de las capacidades del sensor y de la credibilidad de las observaciones:** es deseable poder estimar cuán capaz es un sensor de proveer observaciones lo suficientemente reales como para usarlas para apoyar en la toma de decisiones.
3. **Acceso a parámetros y procesos que permitan el procesado de observaciones por pedido:** es deseable la capacidad de soportar la geolocalización de las observaciones por pedido así como el procesado de las observaciones por software genérico, sin la necesidad de conocer el sistema de sensores de antemano.
4. **Recuperación de observaciones en tiempo real o en intervalos de tiempo en codificaciones estándares:** es deseable acceder e inmediatamente utilizar observaciones de nuevos sensores descubiertos en herramientas de toma de decisiones, modelos y simulaciones sin la necesidad de desarrollar lectores específicos para cada

sensor.

5. **Asignación de tareas a sensores y simuladores:** es deseable asignar tareas a sensores o simuladores y proveer una interfaz común para estas.
6. **Subscripción y publicación de alertas basadas en las observaciones de sensores o simulaciones:** es deseable proveer de una vía por la cual un sensor o simulación pueda alertar de problemas, basándose para ello en ciertos criterios; de igual forma es deseable poder inscribirse a estas alertas. Los criterios pueden tener cualquier nivel de complejidad.

El estándar SWE se ha aplicado en varios proyectos de investigación y los sistemas muestran su viabilidad e idoneidad en aplicaciones del mundo real. Estas aplicaciones van desde el sistema de monitoreo de flujo de escombros para Taiwán[8], la utilización de los SWE en sistemas de gestión de desastres [9] [10] [11], a un sistema de alerta contra los tsunamis en Indonesia [12]. Sin embargo, se requiere un esfuerzo considerable para hacer que los sensores y sus observaciones estén disponibles en un SW, ya que los métodos y mecanismos para automatizar este proceso no son abundantes.

El estándar SWE ha sido construido por miembros de OGC incluyendo las siguientes especificaciones (algunas de las cuales serán explicadas luego con más detalle):

1. **Sensor Model Language (SensorML, en español Lenguaje de Modelado de Sensores):** Modelos estándar y de esquema XML para la descripción de los procesos dentro de los sistemas de procesamiento de sensores y de observaciones, incluye modelos de representación de datos comunes válidos para todas las codificaciones SWE y estándares de interfaz de servicio; proporciona la información necesaria para el descubrimiento, georeferenciación, y el procesamiento de las observaciones, así como sensores de asignación de tareas y simulaciones.
2. **Observations & Measurements (O&M, en español Observaciones y Medidas):** Los modelos generales y codificaciones XML para observaciones y mediciones hechas

usando sensores [13].

3. **Transducer Model Language (TML, en español Lenguaje de Modelado de Traducciones):** Aproximación conceptual y codificación XML para soportar transmisión en tiempo real de las observaciones y tareas de comandos desde y hacia el sistema de sensores (SW) [14].
4. **Sensor Observation Service (SOS, en español Servicio de Observación de Sensores):** Una interfaz abierta para un servicio por el cual un cliente puede obtener observaciones y descripciones de los sensores y de la plataforma de uno o más sensores [14].
5. **Sensor Planning Service (SPS, en español Servicio de Planificación de Sensores):** Una interfaz abierta para un servicio por el cual un cliente puede determinar la viabilidad de la recogida de datos de uno o más sensores o modelos y presentar solicitudes de acceso a observaciones de estos sensores y procesos configurables [14].
6. **Sensor Alert Service (SAS, en español Servicio de Alertas de Sensores):** Una interfaz abierta para un servicio web para la publicación y la suscripción para entregar las alertas de los sistemas de sensores o de simulación [14].
7. **Web Notification Service (WNS, en español Servicio de Notificaciones Web):** Una interfaz abierta para un servicio por el cual un cliente puede realizar diálogos asíncronos (intercambia mensajes) con uno o más servicios [14].

La infraestructura de los estándares SW, definida por estas especificaciones, constituye una vía para el descubrimiento, evaluación y control de las fuentes de datos en vivo y datos del sensor archivados.

1.2.1 Lenguaje de Modelado de Sensores

La medición de un fenómeno que tiene como resultado una observación, consiste en una

serie de procesos, comenzando con el muestreo y detección, y seguido tal vez por la manipulación de datos. La división entre mediciones y posprocesamiento se ha vuelto más borroso con la aparición de sensores más complejos e inteligentes. Por esto SensorML define modelos y esquemas de XML para describir procesos, incluyendo la medición de un sistema de sensores y el post-procesado de las mediciones. SensorML soluciona ciertas necesidades en la comunidad de sensores, como [15]:

1. **Revelación de sensores, sistemas de sensores y procesos:** SensorML es un medio que permite a sensores, sistemas de sensores y procesos hacerse conocer. Este estándar provee de una rica colección de metadatos que puede ser empleada para estos fines.
2. **Procesado de Observaciones a petición:** Las cadenas de procesos de geolocalización o procesado de observación de nivel superior pueden ser descritas en SensorML, reveladas y distribuidas por la web y ejecutadas a petición sin un conocimiento previo de las características del sensor o el procesador.
3. **Calificación de las Observaciones:** SensorML puede proveer una descripción completa y clara (no ambigua) de la calidad de una observación. O sea, puede describir detalladamente el proceso de donde viene una observación desde su adquisición por uno o más detectores hasta su procesado, quizás incluso su interpretación por un analista.
4. **Apoyo para la asignación de tareas, la observación y los servicios de alerta:** Las descripciones de los sistemas de sensores o simuladores en SensorML pueden ser empleados para dar apoyo y establecer SOS, SPS y SAS. En él se definen datos que son usados a lo largo de SWE.
5. **Autoconfiguración y redes autónomas de sensores:** SensorML permite el desarrollo de sensores, simulaciones y procesos *plug and play*[2], que coherentemente se conecten a sistemas de soporte de decisiones. Sus características autodescriptivas

2 plug and play: Término empleado en la informática para designar componentes que pueden conectarse a sistemas sin necesidad de configuración.

posibilitan también ser añadido a redes de sensores autoconfiguradas.

6. **Almacenamiento de parámetros de sensores:** Finalmente, SensorML provee un mecanismo para almacenar los parámetros fundamentales sobre sensores y procesos, de modo que las observaciones de esos sistemas puedan ser procesadas y mejoradas en tiempo después de que sean ejecutadas. Esto suele ser crucial para aplicaciones de amplio rango de acción, como monitores de cambio global.

1.2.2 Observaciones y Medidas

La funcionalidad general de un sensor es observar una o más propiedades en su ambiente determinado. Cada una de las cuales pertenece a una característica que representa un objeto identificable. De ahí que O&M consta de dos partes [16]:

1. **Esquema de la Observación:** Describe un modelo conceptual y una codificación para observaciones y medidas, este está formalizado como un esquema de aplicación pero puede ser empleado en varias aplicaciones.
2. **Características de la muestra:** Describe un modelo conceptual y una codificación para la propiedad a observar.

Según O&M, cada propiedad observada pertenece a una propiedad de interés. Incluso cuando muchas veces sean tratadas idénticamente, existe una diferencia conceptual entre le propiedad observada y la propiedad de interés por ejemplo: en un centro meteorológico en el que se mide la temperatura la propiedad de interés es la temperatura de la región donde está este, pero la propiedad observada realmente es la temperatura en las cercanías del termómetro.

O&M define una observación como el hecho de medir una propiedad o fenómeno para, a partir de ello, estimar su valor. Una observación emplea un procedimiento para determinar el valor de su resultado, que puede involucrar un sensor, procesamiento analítico, simulación y otras herramientas matemáticas. Este proceso será normalmente descrito en el SensorML.

De esta forma la estructura de una observación en O&M describe la propiedad de interés, el proceso aplicado, la propiedad observada, el resultado y la calidad de este, el tiempo de muestreo, el instante de tiempo en que el resultado fue obtenido, metadatos adicionales y parámetros que describen la observación aun cuando no estén directamente relacionados con el proceso o la propiedad de interés.

O&M define las siguientes especializaciones de la observación genérica:

1. **CountObservation:** Para propiedades de un solo valor la observación en las que el resultado es un entero representando la cantidad de la propiedad observada.
2. **CategoryObservation:** Si el resultado es textual de un vocabulario controlado (una categoría).
3. **TruthObservation:** Si el resultado es un valor lógico (normalmente cuando se observa la existencia de una propiedad).
4. **GeometryObservation:** El resultado es geométrico.
5. **TemporalObservation:** El resultado es un objeto temporal.
6. **ComplexObservation:** El resultado representa un fenómeno multicomponente.
7. **Measurement:** Es el resultado directo de una medición, este está descrito usando un valor en una escala.

Por otro lado, si la propiedad varía en el espacio-tiempo el tipo de observación es modelado como un campo de acción discreto de tipo:

1. **PointCoverageObservation:** Si el resultado es un punto de acción con propiedades muestreadas dentro de la propiedad de interés.
2. **DiscreteCoverageObservation:** Si el resultado es un campo de acción discreto general.
3. **TimeSeriesObservation:** Si el resultado es un instante de tiempo que muestrea una

característica de la propiedad de interés en diferentes momentos.

4. **ElementCoverageObservation:** Si el resultado es un grupo de elementos que hacen referencias a objetos en algún sitio que proveen información geométrica de la propiedad de interés.

1.2.3 Sensor Observation Service (SOS)

El principal objetivo de SOS es proveer acceso a las observaciones de los sensores web en una forma estándar que sea consistente para todos los sistemas de sensores incluyendo remotos, fijos y sensores móviles. SOS ha aprovechado las especificaciones de las Mediciones y Observaciones (O&M) para modelar observaciones de sensores y las especificaciones de SensorML para modelar sensores y sistemas de sensores.

Un SOS organiza colecciones de observaciones del sistema sensor en ofertas de observaciones. Estas son grupos de elementos sin superposición de observaciones relacionadas, cada una se ve limitada por una serie de parámetros, entre ellos los siguientes: periodo(s) de tiempo en que puede solicitarse la información, fenómeno que se está detectando, región geográfica que contiene el sensor y región geográfica que es objeto de las observaciones de los sensores.

1.2.4 Implementaciones del servidor SOS sobre estándares OGC-SWE

IstSOS: Es una implementación del servidor SOS por OGC y escrita en Python. Provee una interfaz gráfica de usuario para permitir el uso sencillo de sus operaciones y un API Rest para automatizar los procedimientos de administración. Es implementado bajo Licencia Pública General (en inglés General Public License, GPL) y se ejecuta en todas las plataformas (Windows, Linux, Mac OS X) a pesar que las pruebas fueran llevadas a cabo en sistemas Linux. IstSOS implementa los 7 estándares de OGC-SWE y acorde a la versión 1.0 del SOS tiene implementado 3 principales interfaces del núcleo de servidor SOS [17].

1. GetCapabilities.

2. DescribeSensor.
3. GetObservation.

52º North: Es una implementación del servidor de SOS escrita por OGC en Java bajo la licencia GPL 2 y se ejecuta en todas las plataformas (Windows, Linux, Mac OS X). Implementa los 7 estándares de OGC-SWE y la versión 2.0 de SOS. La versión 2.0 de SOS plantea 13 interfaces de las que tiene implementada 7 [18] :

1. GetCapabilities (implementado).
2. GetObservation (implementado).
3. DescribeSensor (implementado).
4. GetFeatureOfInterest (implementado).
5. GetObservationById.
6. InsertSensor (implementado).
7. UpdateSensorDescription.
8. DeleteSensor.
9. InsertObservation (implementado).
10. InsertResultTemplate.
11. InsertResult.
12. GetResultTemplate.
13. GetResult (implementado).

Otras características que son soportadas:

1. Codificación WaterML 2.0.
2. RESTFul.
3. DeleteObservation operación para eliminar observaciones a partir del identificador (no

es parte de las especificaciones de SOS 2.0).

4. Perfil hidrológico para el Servicio de Observación de Sensores 2.0.

A partir de las características presentadas se puede resumir que ambos implementan los estándares de OGC para SOS. ItsSOS está implementado en Python mientras que 52north en Java y ambos se ejecutan en todas las plataformas (Window, Linux, Mac OS X). Ambas presentan servicio REST[3] para la automatización de la administración. Pero ItsSOS implementa la versión 1.0 del estándar SOS mientras que 52north la versión 2.0. 52north implementa 7 interfaces de las 13 que presenta OGC para SOS versión 2 y también presenta 4 características extras que apoyan en la flexibilidad de los procedimientos presentados por OGC para SOS.

Por tanto se puede concluir que 52north implementa la versión 2.0 de OGC para SOS que es la versión estable de los estándares SWE. Contiene características extras las cuales no están en los estándares OGC pero proporcionan una mayor flexibilidad que estos para SOS.

1.3 Selección de Tecnología y Metodología de desarrollo

Las metodologías imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente. Lo hacen desarrollando un proceso detallado con un fuerte énfasis en planificar inspirado por otras disciplinas de la ingeniería. La metodología es un proceso que define quien debe hacer qué, cómo y cuándo debe hacerlo.

3 *REST:* Representational State Transfer: transferencia de estado representacional.

1.3.1 Proceso Unificado Ágil

El desarrollo de software no es una tarea fácil. Prueba de ello es que existen numerosas propuestas metodológicas que inciden en diferentes dimensiones del proceso de desarrollo. Por una parte tenemos aquellas propuestas más tradicionales que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán. Estas propuestas han demostrado ser efectivas y necesarias en un gran número de proyectos, pero también han presentado problemas en otros muchos. Una posible mejora es incluir en los procesos de desarrollo más actividades, más artefactos y más restricciones, basándose en los puntos débiles detectados. Sin embargo, el resultado final sería un proceso de desarrollo más complejo que puede incluso limitar la propia habilidad del equipo para llevar a cabo el proyecto. Otra aproximación es centrarse en otras dimensiones, como el factor humano o el producto software. Esta es la filosofía de las metodologías ágiles, las cuales dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Este enfoque está mostrando su efectividad en proyectos con requisitos muy cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad. Las metodologías ágiles están revolucionando la manera de producir software, y a la vez generando un amplio debate entre sus seguidores y quienes por escepticismo o convencimiento no las ven como alternativa para las metodologías tradicionales.

Aunque no existe una metodología general para hacer frente con éxito al desarrollo de software, se escoge como metodología de desarrollo de software a AUP debido que la Universidad de las Ciencias Informáticas cuenta actualmente con 14 centros productivos empleando el uso de diferentes metodologías de desarrollo entre estas robustas y ágiles. El factor determinante de esta metodología radica en la forma de como planificar el proyecto y de la estimación de tiempo. Factor determinante en la culminación exitosa de todo desarrollo de software. Para lograr erradicar el problema detectado la Universidad de las Ciencias

Informáticas propuso una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la Universidad [19].

1.3.2 Lenguajes de Programación

Python es un lenguaje de programación interpretado, fuertemente tipado y de tipado dinámico; diseñado por Guido Van Rossum, cuya principal característica es la legibilidad [20]. Su intérprete más comúnmente empleado (CPython) está implementado en C y se desempeña con un consumo de memoria y velocidad en el rango del resto de los intérpretes que usan el mecanismo GIL[4] (PHP, Ruby, Perl entre otros), incluso en muchas pruebas para evaluar rendimiento obtiene mejores resultados que el resto de ellos. El lenguaje tiene un sinnúmero de bibliotecas de terceros que le permiten interactuar con una inmensa cantidad de protocolos, bases de datos, procesar formatos de texto plano y archivos binarios; en muchos casos estas (las que requieren velocidad o poco consumo de memoria) están implementadas en C o C++, así que realmente el desempeño de las aplicaciones no se ve afectado por el lenguaje, ya que este solo funge como conductor de la lógica.

Por otra parte Python tiene características ventajosas como:

- Ser muy legible (suele ser catalogado por muchos como pseudocódigo que se puede ejecutar), cuestión que facilita el desarrollo si se toma en cuenta que el código se lee más veces de las que se escribe.
- Ser multiparadigma, entre los paradigmas en el que puede ser empleado está la Programación Orientada a Objetos, pues todos sus tipos son clases a diferencia de otros lenguajes (como Java y C++) donde hay tipos base y clases (por eso en java hay int e Integer).

Para el desarrollo de la solución se seleccionó la versión 2.7 de este lenguaje pues esta, está presente en la mayoría de las distribuciones de Linux, así como en las de MacOS.

4 Global Interpreter Lock: Mecanismo empleado en informática para sincronizar la ejecución de hilos de modo que estos se ejecuten solo uno a la vez.

1.3.3 Lenguaje de modelado

Lenguaje Unificado de Modelado (siglas en inglés UML): Estándar que se ha adoptado a nivel internacional por numerosos organismos y empresas para crear esquemas, diagramas y documentación relativa a los desarrollos de software. Objetivamente se puede planificar y documentar cómo se construyen los programas informáticos, ayudando así al proceso de gestión de la construcción del software.

1.3.4 Entorno de desarrollo

Como entorno de desarrollo se eligió Sublime Text 3, este editor se caracteriza por tener un bajo consumo de recursos, un alto rendimiento, una interfaz fluida y una alta extensibilidad. De modo que existen para él paquetes como:

1. **Anaconda:** Este paquete mejora el soporte de Python, gracias a que brinda funcionalidades como el completamiento inteligente (hace introspección en el código para determinar tipos) de código, chequeo estático, autocorrección de estilo de código, navegación de código (permite ir hasta la definición de clases y funciones), entre otras.
2. **Stupid Ident:** Configura automáticamente el sangrado del editor en dependencia de la extensión del archivo que se esté modificando.
3. **Terminal:** Permite la ejecución de un emulador de terminal (comúnmente conocido como consola) desde el editor de texto, siendo este ejecutado en la dirección del archivo que se modifica.
4. **Character Table:** Permite la inserción de caracteres que no están disponibles en el teclado que se emplea, con solo conocer su nombre, su valor unicode o un dígrafo equivalente, por ejemplo: puede escribirse ℃ usando el dígrafo oC.

Estas características sumadas a un diseño de interfaz agradable, hacen de Sublime un editor potente y a la vez simple de usar.

1.3.5 Biblioteca lxml

La reutilización de código permite economizar tiempo y reducir redundancia. De esta

forma se aprovechan trabajos anteriores. Se reutiliza código copiando parcial o totalmente desde el programa antiguo al programa actual, pero la mejor forma de reutilizar código es dejando el código reusable en su lugar de origen y llamándolo desde los diferentes programas. Dando lugar a las bibliotecas de software en las que se agrupan varias operaciones comunes para facilitar el desarrollo de programas nuevos.

De modo que se decidió emplear la biblioteca lxml para lectura y escritura de XML. Esta emplea internamente las bibliotecas de libxml2 y libxslt (estas dos escritas en C). Esto posibilita que combine la velocidad que brindan estas bibliotecas con una interfaz programable compatible con la de ElementTree, la cual es una de las bibliotecas estándares del lenguaje Python para estos menesteres.

Otra característica de esta biblioteca es que posibilita realizar búsquedas dentro de un XML empleando consultas XPath[5], lo cual facilita la tarea de encontrar atributos y elementos específicos en un documento con este formato.

1.3.6 Herramienta para realizar pruebas Pytest

Pytest es una herramienta diseñada para realizar pruebas unitarias a código escrito en python. Esta tiene entre sus características [21]:

- Que puede ser empleada en cualquier sistema operativo Posix[6] y en Windows y con cualquier versión de python entre 2.6 y 3.4.
- Que es integrable con otros sistemas de pruebas.
- Que puede ser extendido de modo que puede agregársele otros tipos de pruebas, un ejemplo es la extensión pytest-cov para realizar pruebas de cobertura.

1.3.7 Sistema de control de versiones

Para facilitar el desarrollo en equipos de trabajo y lograr la integración correcta que

5 XPath: lenguaje para escribir consultas que indican como buscar determinado elemento o atributo en un documento XML.

6 Posix: acrónimo de portable operating system.

permita a cada integrante del equipo de desarrollo crear, copiar y borrar las carpetas del proyecto en desarrollo con la misma flexibilidad con la que lo haría si estuviese en su disco duro local; se utiliza el Subversion (SVN). Esta es una herramienta de control de versiones de código libre basada en un repositorio cuyo funcionamiento se asemeja enormemente al de un sistema de ficheros. Utiliza el concepto de revisión para guardar los cambios producidos en el repositorio. Entre dos revisiones sólo guarda el conjunto de modificaciones (delta), optimizando así al máximo el uso de espacio en disco [22].

Conclusiones parciales

La investigación realizada sobre el objeto de estudio, así como sobre las metodologías y herramientas para darle solución al problema de la investigación; permitió seleccionar las más adecuadas. Además se identificó la necesidad de implementar los estándares SensorML y O&M como vía para lograr la comunicación con redes de sensores web. Se seleccionó el servidor 52ºNorth para publicar la información brindada por los sensores.

Capítulo 2: Análisis y diseño de la solución

Introducción

El presente capítulo refleja las actividades realizadas en los procesos de análisis y diseño de la solución propuesta. Se exponen los artefactos más importantes que describen el flujo normal del sistema, tales como especificaciones de requisitos no funcionales, historias de usuarios, diagramas y las principales clases que componen el mecanismo, reflejando así, la solución en detalles.

La habilitación de sensores web es una tecnología que permite compartir los datos medidos por sensores de diversos proveedores, así como sus metadatos, en un lenguaje común; de modo que los mismos pueden ser consumidos y procesados de una forma transparente al sensor en cuestión. La gran cantidad de sensores que presentan los sistemas SCADA-GALBA los convierte en una fuente de información que mezclada con sensores de otros sistemas podría ser aprovechada para evitar accidentes industriales o apoyar la toma de decisiones empresariales; generando estos beneficios a partir de combinar la información de diversas fuentes, como permitiría el empleo de una habilitación de sensores web.

El presente trabajo describe el proceso de construcción de una biblioteca para desarrollar sensores web que permitan compartir los datos recolectados por el sistema SCADA-GALBA.

2.1 Roles de la Metodología

AUP define roles en los cuales se desempeñan cada uno de los que de una forma u otra están vinculados con el proceso de desarrollo o en interacción con el mismo.

Persona relacionada con la biblioteca	Responsabilidad
Desarrollador	Persona encargada de implementar la biblioteca.
Probador	Persona encargada de probar el funcionamiento de la biblioteca.

Tabla 1: Roles de AUP.

2.2 Especificación de requisitos

Tras una entrevista con el cliente, fueron recogidos la especificación de requisitos de software. Esta es una descripción completa de las necesidades del producto. Compuesta por el conjunto de requisitos funcionales y no funcionales, todos los cuales deben ser reflejados en ella, con una redacción clara; de modo que no se pueda mal interpretar. Los requisitos deben estar organizados jerárquicamente según su nivel de relevancia para el negocio.

2.2.1 Requisitos funcionales

Entre los artefactos definidos en la metodología se encuentran los requisitos funcionales, que serán representados en Historias de Usuarios (HU), estas son modelos que representan una breve descripción del comportamiento del sistema. Esta herramienta sirve para dar a conocer al equipo de desarrollo cada requisito del sistema. Se puede considerar que las HU juegan un papel similar a los casos de uso de otras metodologías. Su tratamiento es dinámico y flexible, lo que permite que en cualquier momento se puedan remplazar por otras más específicas o generales, añadirse nuevas o ser modificadas.

Las historias de usuario tienen, los siguientes atributos:

Nombre de la historia: Atributo que contiene el nombre de la HU.

Usuario: El usuario del sistema que utiliza o protagoniza la HU.

Prioridad en el negocio: Evidencia el nivel de prioridad de la HU en el negocio. Se considera Alta en caso de que la HU sea imprescindible en el negocio, Media en caso de que

su realización o no lo afecte considerablemente y Baja cuando no se considera una prioridad para el negocio.

Riesgo de desarrollo: Evidencia el nivel de riesgo en caso de no realizarse la HU. Se considera alta, cuando el riesgo de no realizar la HU implica en el funcionamiento de la plataforma. Media cuando el riesgo de no realizarla es medianamente importante y baja en caso de que no se considere un riesgo el hecho de tardar en la realización de la HU y no implique cambios en el funcionamiento de la plataforma.

Estimación de esfuerzo: Este atributo es una estimación hecha por el equipo de desarrollo del tiempo de duración de la HU. Cuando el valor es 1 equivale a una semana ideal de trabajo tomando como semana ideal una de 5 días hábiles con 40 horas de trabajo, es decir, 8 horas diarias. Por lo que cuando el valor de dicho atributo es 0.5 equivale a 2 días y medio de trabajo, lo que se traduce en 20 horas.

Iteración asignada: Se especifica la iteración a la que pertenece la HU correspondiente.

Descripción: Posee una descripción de lo que realizará la HU.

Observaciones: Se especifica las observaciones del cliente.

Dependencias: Se especifican las dependencias de la HU con respecto a otras enumerándolas.

Luego de analizar los datos y especificaciones de cada campo comprendido en el modelo de HU, se obtuvo para el presente trabajo las historias que se muestran a continuación:

Historia de Usuario	
Número: 1	**Nombre historia:** Recolectar datos de SCADA-GALBA.
Usuario: Sensor	**Iteración Asignación:** 1
Prioridad en el negocio: Alta	**Riesgo en el desarrollo:** Alto
Estimación de esfuerzo: 3	
Descripción: Los datos del SCADA-GALBA serán extraídos mediante un Servicio Web el cual tiene expuesto un Xml que describe la interfaz del servicio.	
Observaciones: El Xml que describe la interfaz del servicio es un wsdl[7].	
Dependencia: -	

Tabla 2: Historia de Usuario 1

Historia de Usuario	
Número: 2	**Nombre historia:** Serializar datos.
Usuario: Sensor	**Iteración Asignación:** 2
Prioridad en el negocio: Alta	**Riesgo en el desarrollo:** Alto
Estimación de esfuerzo: 5	
Descripción: Los datos recolectados del SCADA-GALBA serán traducidos a O&M estructurado para lograr la interoperabiliad del sistema.	
Observaciones:	
Dependencia: 1	

Tabla 3: Historia de Usuario 2

7 wsdl: siglas en inglés para *Web Service Description Language*, Lenguaje de descripción de servicios web.

Historia de Usuario	
Número: 3	**Nombre historia:** Inscribir Sensor en el SOS.
Usuario: Administrador	**Iteración Asignación:** 3
Prioridad en el negocio: Alta	**Riesgo en el desarrollo:** Alto
Estimación de esfuerzo: 2	
Descripción: A través del protocolo HTTP el sensor debe inscribir el SensorML en el SOS mediante la interfaz RegisterSensor, para esta identificarse en el sistema. El sensor tendrá como respuesta un identificador para distinguirse entre los otros sensores.	
Observaciones: -	
Dependencia: 1, 2	

Tabla 4: Historia de Usuario 3

Historia de Usuario	
Número: 4	**Nombre historia:** Enviar datos al SOS.
Usuario: Sensor	**Iteración Asignación:** 3
Prioridad en el negocio: Alta	**Riesgo en el desarrollo:** Alto
Estimación de esfuerzo: 2	
Descripción: El Sensor debe enviar los datos al SOS mediante la interfaz InsertSensor, la cual permite a partir del identificador del sensor y el xml serializado en el estándar O&M insertar los valores en el SOS.	
Observaciones: -	
Dependencia: 1, 2, 3	

Tabla 5: Historia de Usuario 4

2.2.2 Requisitos no funcionales

Características de Software:

La biblioteca puede ser usada tanto en Linux, Windows o MacOS. Los cuales tienen que tener instalado Python 2.7.

Características de Hardware:

Es recomendado como mínimo:

- 256MB de RAM.
- Intel(R) Celeron(R) CPU G1830 @ 2.80GHz.

2.3 Estimación de esfuerzo por Historias de Usuarios

Con el fin de conocer el tiempo aproximado del ciclo de vida de la HU se realizó una estimación del esfuerzo de las historias de usuarios identificadas. Obteniéndose de esta, los resultados expuestos en la tabla siguiente:

Historias de Usuario	Puntos de estimación
Recolectar datos de SCADA	3 semanas
Serializar datos	3 semana
Inscribir el Sensor en el SOS	2 semana
Enviar Datos al SOS	3 semana
Total	11 semanas

Tabla 6: Historias de Usuario

2.3.1 Plan de duración de las iteraciones

En la metodología AUP la fase de ejecución se divide en iteraciones para facilitar el desarrollo. Para cada iteración se define un módulo o un conjunto de HU que se van a implementar. Al final de cada iteración se tiene como resultado la entrega del módulo correspondiente el cual debe haber superado las pruebas de aceptación que se establecieron para verificar el cumplimiento de los requisitos. Las tareas que no se realizaron en una iteración se tienen en cuenta para la próxima.

Las historias de usuarios fueron implementadas en orden de las dependencias entre ellas.

Iteración	Orden de las HU para implementar	Puntos de estimación
1	HU #1 Recolectar datos de SCADA-GALBA	3 semanas
2	HU #2 Serializar datos	3 semana
3	HU #3 Inscribir Sensor en el SOS HU #4 Enviar datos al SOS	5 semanas

Tabla 7: Iteraciones

2.3.2 Plan de entregas

El Plan de entrega (PE) especifica qué HU alcanza el fin del ciclo de vida de su implementación, de este modo se permite conocer con exactitud cuáles HU serán desarrolladas en las próximas entregas. El Plan de entrega es negociado con el cliente y el equipo de desarrollo durante las reuniones de Plan de entrega.

Plan de Entregas				
Fecha de Reunión de Planificación:				
Nombre del Documentador	Jorge Luis Acosta, Leiser Fernández Gallo			
Entrega No. 1				
Historias de usuarios a implementar en las Entregas				
No. HU	Titulo	Prioridad	Fecha de entrega	Liberación en la que se incluirá
1	Recolectar datos de SCADA-GALBA	Alta	5/02/2015	1
2	Serializar datos	Alta	26/02/2015	2
3	Inscribir Sensor en el SOS	Alta	12/03/2015	3
4	Enviar datos al SOS	Alta	2/04/2015	4

Tabla 8: Plan de Entregas

2.4 Diseño de la solución propuesta

Dada la flexibilidad de la metodología empleada para el desarrollo de la solución es posible emplear varias formas de representación de las clases que componen el sistema. Haciendo uso

de las tarjetas de contenido, responsabilidad y colaboración (CRC) serán representadas las que conforman la solución. También se empleará un diagrama de clases que servirá para expresar las relaciones entre ellas de una forma gráfica.

2.4.1 Tarjetas CRC

Las tarjetas CRC permiten desprenderse del método de trabajo basado en procedimientos y trabajar con una metodología basada en objetos. Debido a la facilidad de uso y entendimiento que proporciona dichas tarjetas, se decidió utilizarlas para diseñar la aplicación.

Tarjeta CRC	
Clase: Sensor	
Superclase: -	
Subclase: -	
Responsabilidad	**Colaboración**
1. Serializa los metadatos del Sensor al estándar SensorML. 2. Serializa las observaciones de los sensores físicos al estándar O&M	Clase: _BaseType

Tabla 9: Tarjeta CRC de la clase Sensor

Tarjeta CRC	
Clase: Publisher	
Superclase: -	
Subclase: -	
Responsabilidad	**Colaboración**
1. Recolectar los datos de la fuente que los proporciona. 2. Crear conexión con el SOS utilizando los metadatos del sensor codificados en el estándar SensorML. 3. Enviar datos al SOS utilizando los datos obtenidos en el estándar O&M.	Clase: Sensor Clase: DataProvider

Tabla 10: Tarjeta CRC de la clase Publisher

Tarjeta CRC	
Clase: _BaseType	
Superclase: -	
Subclase: Count, Text, Boolean, Category, Quantity, DataArray, DataRecord	
Responsabilidad	**Colaboración**
1. Describir la interfaz a implementar por cada una de sus subclases. 2. Proveer mecanismos genéricos para la codificación de tipos.	Módulo: XML

Tabla 11: Tarjeta CRC de la clase _BaseType

El resto de las tarjetas está recogido en el anexo correspondiente.

2.4.2 Patrones de diseño

Estilos y patrones de diseño: Describen una clase de arquitectura o pieza identificada. Los estilos se manifiestan en arquitectura teórica descriptiva a alto nivel de abstracción mientras que los patrones se identifican por todo el programa, siendo estos los que definen la arquitectura a un nivel práctico.

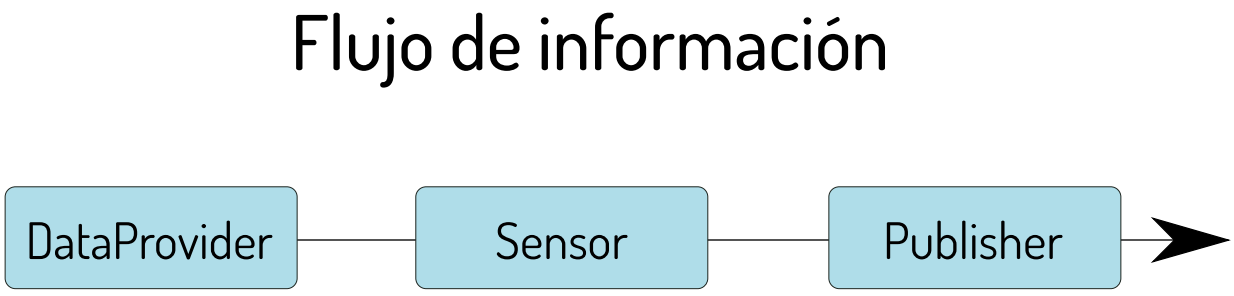


Figura 3: Tuberías y Filtros

Se utilizó el estilo **Tubería y Filtro** ya que este encaja en la arquitectura de flujo de datos. Una tubería (pipeline) es una popular arquitectura que conecta componentes computacionales (filtros) a través de conectores (pipes) de forma que las operaciones se manejan de forma de un flujo.

Patrones Grasp (General Responsibility Assignment Software Pattern):

Patrones de software para asignar responsabilidades, son considerados como "Buenas Prácticas" para el diseño del software. El objetivo fundamental de utilizarlos es caracterizar objetos y clases. Conocer a quien asignar una responsabilidad, quien es el responsable y quien puede manejarlas [23].

1. **Experto en información:** Asignar la responsabilidad al experto en información, la clase que tiene la información necesaria para llevar a cabo la responsabilidad.
2. **Creador:** Asignar la responsabilidad de crear nuevas instancias. Crear nuevas instancias por clases que: tengan la información necesaria para realizar la creación de objetos, use directamente las instancias creadas del objeto, almacene o maneje varias instancias de objetos o contiene o agrega a la clase.
3. **Controlador:** Asigna responsabilidades de gestión de eventos en el sistema a una clase que represente, el sistema global, dispositivo o subsistema (controlador de

fachada) o represente un escenario de casos de usos en el que tiene lugar el evento del sistema (controlador de casos de uso o de sesión).

4. **Alta cohesión:** Asignar responsabilidades de manera que la información que almacena la clase sea coherente y esté relacionada con la clase. Logrando mantener un nivel de complejidad manejable.
5. **Bajo acoplamiento:** Asignar responsabilidades a las clases que dependen entre sí, de modo que estén lo menos ligadas posibles, de forma tal que se entienda la mayor parte de su estructura y sus funcionalidades.
6. **Polimorfismo:** Asignar responsabilidad del comportamiento (utilizando operaciones polimórficas) a los tipos para los que varía el comportamiento.

Patrones Gof (Gang of Four):

- **Decorador:** se encarga de añadir directamente funcionalidades a un Objeto, permitiendo no tener que heredar sucesivamente de una clase para agregar la funcionalidad. En la biblioteca este patrón se empleará para crear propiedades estáticas a partir de los métodos de instancia. Para decorar métodos y añadir a estos ciertas funciones (gracias a que en python los métodos son también clases) que los hacen ser vistos desde el resto del sistema como atributos (lo que se conoce en python como descriptor); pero a diferencia de property (en español: propiedad), decorador que provee la biblioteca estándar del lenguaje para estos menesteres, el método se ejecuta una sola vez y luego su resultado es guardado.
- **Estrategia:** se clasifica como patrón de comportamiento. Permite mantener un conjunto de algoritmos de entre los cuales el objeto cliente puede elegir aquel que le conviene o intercambiarlo dinámicamente según su necesidad. En la biblioteca este patrón se empleará para delegar la serialización de cada tipo de dato por parte del sensor, lo cual posibilita una mayor flexibilidad (pues podrían añadirse nuevos tipos de

datos posteriormente sin si necesidad modificar la clase sensor) así como separar las responsabilidades.

- **Iterador** se clasifica como un patrón de comportamiento. Provee la forma de acceder a elementos de un objeto agregado secuencialmente sin exponer la estructura del objeto. En la biblioteca este patrón se empleará para abstraer al resto del sistema de la forma en que el proveedor de datos, obtiene estos cada vez que sea requerido.

2.4.3 Diagrama de clases

Los diagramas de clases son un tipo de diagrama estático que describe la estructura de un sistema mostrando las clases del sistema, sus atributos, operaciones (métodos) y las relaciones entre los objetos.

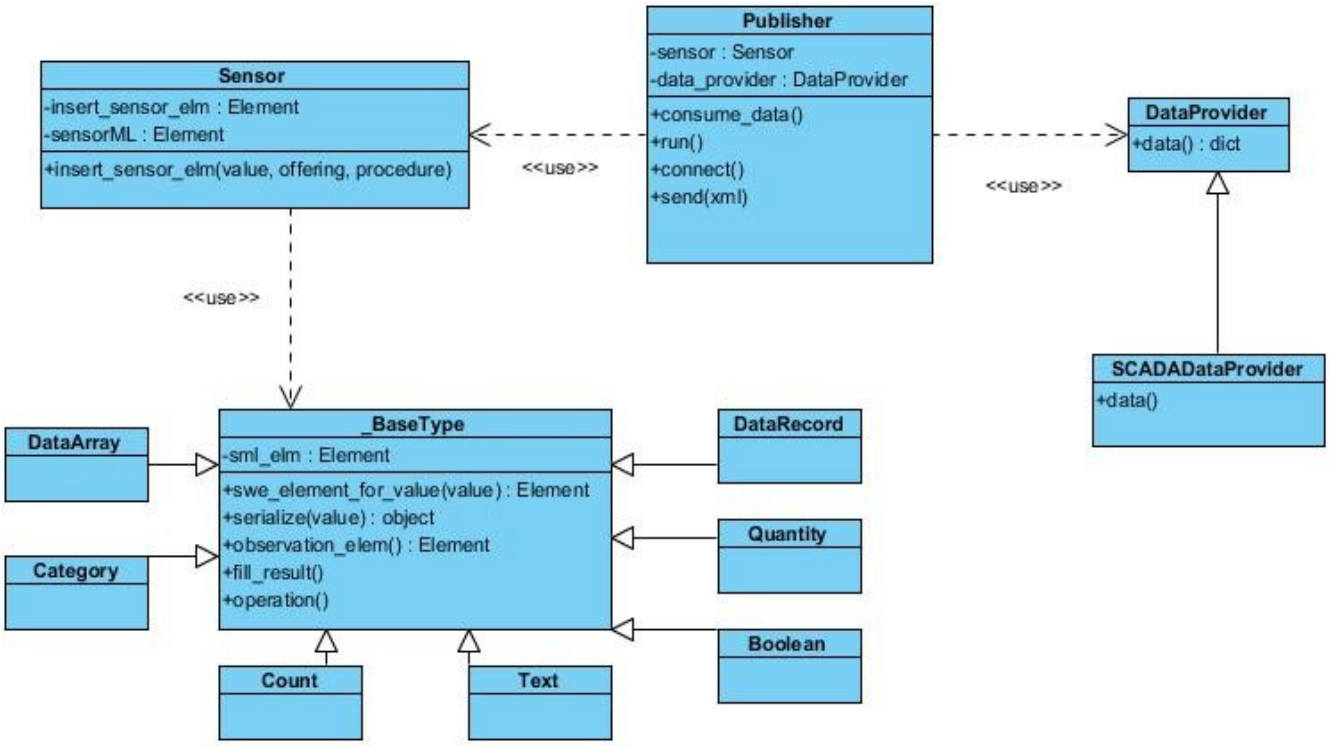


Figura 4: Diagrama de Clases

La Figura 4 representa el diagrama de clases de la biblioteca, este está compuesto por un conjunto de clases de las cuales *_BaseType* y *DataProvider* son abstractas.

_BaseType es la clase padre de todos los tipos de datos empleados para la serialización del

SensorML y el O&M. Heredan de ella siete clases, cada una de las cuales implementa la forma en que se codifica cada uno de los tipos en estos dos estándares: la clase *DataArray* codifica elementos de tipo array; *Category*, elementos que pueden tomar valores de un nominales de un conjunto definido; *Count*, elementos de tipo entero no negativo; *Text*, elementos de tipo cadena de caracteres; *Boolean*, elementos de tipo lógico (verdadero o falso); *Quantity*, elementos de tipo real y *DataRecord*, elementos de tipo mapa o diccionario (esto es que tienen estructura llave-valor).

La clase *Publisher* delega dos funciones fundamentales en las clases *DataProvider* y *Sensor*, las cuales son:

- La clase *DataProvider* provee a la clase *Publisher* los datos extraídos de la fuente, haciendo transparente para *Provider* el mecanismo de extracción de datos.
- La clase *Sensor* provee a *Provider* los datos codificados en el estándar O&M y los metadatos del sensor en el estándar SensorML, después de haberlos recibos de la clase *Provider*.

A partir de las dos funciones anteriormente explicadas la clase *Provider* es capaz de inscribir el sensor en el SOS, enviando el SensorML y de publicar las observaciones del sensor en el SOS, enviando el O&M, ambas operaciones a través del protocolo SOAP.

La clase *Sensor* se encarga de proveer el SensorML y O&M a la clase *Publisher*. Haciendo uso de las clases hijas de *_BaseType* quienes se especializan en la serialización de elementos de un tipo de dato determinado.

La clase *DataProvider* es una clase abstracta, por lo cual es necesario heredar e implementar el método *data*. Este debe devolver en cada ocasión los datos extraídos de la fuente de datos en un diccionario, dando así la posibilidad de extracción de datos de diferentes fuentes y manteniendo la transparencia del proceso de extracción para el resto de las partes de la biblioteca.

La clase *SCADADataProvider* hereda de la clase DataProvider implementando el método data, el cual extrae los datos del servicio *SCADA-Comunicación con Terceros*. Esta es un intermediario entre dicho servicio y la clase *Provider*.

Conclusiones parciales

La elaboración de las historias de usuario como descripción de los requisitos funcionales y la aplicación del estilo arquitectónico de tuberías y filtros así como de patrones de diseño, facilitaron el diseño de la solución permitiendo obtener una visión más clara de la misma. Se concluye además, que todos los artefactos generados por la metodología en esta etapa guiarán de forma efectiva el desarrollo de la solución.

Capítulo 3: Implementación y pruebas de la solución propuesta

Introducción

En el presente capítulo se abordarán las tres iteraciones que se concibieron durante la fase de planeamiento, exponiéndose las tareas generadas por cada historia de usuario y las pruebas de aceptación efectuadas sobre el sistema.

3.1 Desarrollo de las iteraciones

En la fase de planificación se desarrollaron las HU correspondientes a las iteraciones a desarrollar. Durante el desarrollo de las iteraciones, se realizaron revisiones del plan de iteraciones. De acuerdo al plan las HU se descomponen en tareas asignándolas a un equipo de desarrollo(o una persona) responsable del resultado de la tarea. Aplicando la práctica de pareja en equipo las tareas fueron desarrolladas en pareja. Las tareas fueron descritas en un lenguaje técnico.

A partir de la planificación realizada con anterioridad el sistema será desarrollado en 3 iteraciones, obteniendo como finalidad una biblioteca con las características anteriormente propuestas.

3.1.1 Iteración 1

En está iteración se da cumplimiento a la HU 1 considerada la HU por donde se debe comenzar a desarrollar el producto, pues el uso del estilo **Filtros y Flujos** la información viaja por un conjunto de filtros que tienen dependencia unos de otros. La HU 1: Recolectar Datos, es la HU por donde comienza el flujo de información del sistema.

Historias de Usuarios Abordadas.		
Historias de Usuarios	Tiempo implementación (semanas)	
	Estimación	Real
Recolectar metadatos	3	3

Tabla 12: Iteración 1.Historias de usuarios abordadas.

HU:	
No. de la Tarea: 1.	**No. de la HU:** 1.
Nombre de a tarea: Conectarse al SCADA.	
Tipo de Tarea: Implementación.	**Puntos Estimados:** 4 días.
Fecha de Inicio: 15/01/2015.	**Fecha de fin:** 19/01/2015.
Programador responsable: Jorge Luis Acosta Alonso.	
Descripción: La conexión a SCADA-GALBA es imprescindible, ya que es la fuente de donde se extraerán los datos.	

Tabla 13: Iteración 1. Tarea 1.

HU:	
No. de la Tarea: 2	**No. de la HU:** 1
Nombre de a tarea: Recolectar metadatos de SCADA.	
Tipo de Tarea: Implementación.	**Puntos Estimados:** 3 semanas.
Fecha de Inicio: 19/01/2015.	**Fecha de fin:** 5/02/2015.
Programador responsable: Leiser Fernández Gallo.	
Descripción: Esta funcionalidad proporciona los metadatos de SCADA-GALBA los cuales son necesarios para las próximas configuraciones. Serán usados para conocer los valores/puntos que el SCADA-GALBA está proporcionando.	

Tabla 14: Iteración 1. Tarea 2.

3.1.2 Iteración 2

En esta iteración se implementó la HU 2. Luego de desarrollar la iteración 1, se tiene implementado el punto de acceso de entrada de la información o datos para acceder a los valores y metadatos de los puntos del SCADA-GALBA. La HU 2 permite serializar los datos y metadatos en los estándares SensorML y O&M.

Historias de Usuarios Abordadas.		
Historias de Usuarios	Tiempo implementación (semanas)	
	Estimación	Real
Serializar Datos	3	3

Tabla 15: Iteración 2. Historias de usuarios abordadas.

HU:	
No. de la Tarea: 1	**No. de la HU:** 2
Nombre de a tarea: Definir el Sensor.	
Tipo de Tarea: Implementación	**Puntos Estimados:** 3 días.
Fecha de Inicio: 5/02/2015	**Fecha de fin:** 8/02/2015
Programador responsable: Jorge Luis Acosta Alonso	
Descripción: Definir el Sensor a partir de los metadatos de los puntos de SCADA-GALBA e información proporcionada por el usuario como la geolocalización.	

Tabla 16: Iteración 2. Tarea 1.

HU:	
No. de la Tarea: 2	**No. de la HU:** 2
Nombre de a tarea: General SensorML.	
Tipo de Tarea: Implementación.	**Puntos Estimados:** 1 semana.
Fecha de Inicio: 8/02/2015.	**Fecha de fin:** 15/02/2015.
Programador responsable: Leiser Fernández Gallo.	
Descripción: Generar SensorML a partir de los metadatos del sensor.	

Tabla 17: Iteración 2. Tarea 2.

HU:	
No. de la Tarea: 3	**No. de la HU:** 2
Nombre de a tarea: Recolectar datos.	
Tipo de Tarea: Implementación	**Puntos Estimados:** 4 días.
Fecha de Inicio: 15/02/2015	**Fecha de fin:** 19/02/2015.
Programador responsable: Jorge Luis Acosta Alonso	
Descripción: Recolectar datos a partir de los puntos anteriormente inscritos.	

Tabla 18: Iteración 2. Tarea 3.

HU:	
No. de la Tarea: 4	**No. de la HU:** 2
Nombre de a tarea: General O&M.	
Tipo de Tarea: Implementación	**Puntos Estimados:** 1 semana.
Fecha de Inicio: 19/03/2015.	**Fecha de fin:** 26/02/2015.
Programador responsable: Leiser Fernández Gallo	
Descripción: Generar el O&M a partir de los datos recolectados anteriormente.	

Tabla 19: Iteración 2. Tarea 4.

3.1.3 Iteración 3

En esta última iteración se implementaron las HU 3 y HU 4. Las cuales desempeñan las funcionalidades de inscripción del sensor en el SOS y envíos de datos al SOS.

Historias de Usuarios Abordadas.		
Historias de Usuarios	Tiempo implementación (semanas)	
	Estimación	Real
Inscribir Sensor en el SOS	2	2
Enviar datos al SOS	3	3

Tabla 20: Iteración 3. Historias de usuarios abordadas.

HU:	
No. de la Tarea: 1.	**No. de la HU:** 3.
Nombre de a tarea: Inscribir el Sensor.	
Tipo de Tarea: Implementación.	**Puntos Estimados:** 2 semana.
Fecha de Inicio: 26/02/2015.	**Fecha de fin:** 12/03/2015.
Programador responsable: Leiser Fernández Gallo.	
Descripción: Para inscribir el sensor en el SOS, se enviará el SensorML a la dirección proporcionada por el usuario encapsulada en una petición SOAP.	

Tabla 21: Iteración3. Tarea 1.

HU:	
No. de la Tarea: 2.	**No. de la HU:** 4.
Nombre de a tarea: Enviar datos al SOS.	
Tipo de Tarea: Implementación.	Puntos Estimados: 1 semana.
Fecha de Inicio: 12/03/2015.	Fecha de fin: 02/04/2015.
Programador responsable: Jorge Luis Acosta Alonso	
Descripción: Para enviar datos al SOS, se enviará el O&M y el ID de la inscripción a la dirección proporcionada por el usuario encapsulada en una petición SOAP.	

Tabla 22: Iteración 3. Tarea 2.

3.1.4 Vista de despliegue

Una vez desarrollada la solución será viable su despliegue para lo cual se hará necesario el empleo de un SOS, para su correcto funcionamiento este depende de que se cumplan las siguientes características [24]:

Características de software:

1. Máquina dedicada.
2. Java Runtime Environment (JRE) 7.0 o mayor.
3. Java Servlet-API 2.5 compatible con la aplicación del servidor:
    1. Tomcat 6 o mayor (No usar versión 8.0.8, 7.0.54 y 6.0.41 porque tienen errores en el proceso de instalación).
4. Sistema de Gestión de Base de Datos:
    1. PostgreSQL/PostGIS.
    2. PostgreSQl 9 o mayor.
    3. PostGIS 2.0 o mayor.

Características de hardware:

Almacenamiento:

Los valores presentados son valores promedios a partir de datos experimentados por la compañía 52noth [25]:

Series de tiempo: Una serie de tiempo es un grupo de observaciones que tienen los siguientes parámetros en común:

1. Característica en común.
2. Procedimiento/Sensor.
3. Propiedad observada/Propiedad observable.
4. Tipo de observación, por ejemplo: numérica.

Requerimientos por observaciones:

1,36 * 10^6 observaciones = 750MB.

1,36 * 10^6 observaciones = 786432000B.

1 observación = 578B.

Por tanto una serie de tiempo con frecuencia de 1 observación por minuto requiere un almacenamiento de:

578B * 60 * 24 * 365 = 303796800B = 289MB por **años** por **serie de tiempo**.

Luego de lo cual se puede concluir que por cada serie de tiempo igual a 1 observación por minuto se necesita 289MB por año. Por lo que se puede concluir que la cantidad de memoria física es:

Cantidad de memoria física = cantidad de observaciones por minuto * 289.

Rendimiento: Se puede determinar que a partir de las **series de tiempo** la cantidad de memoria RAM necesaria para enviar la mayor **serie de tiempo** en un momento determinado:

Disponibilidad en RAM (MB) = Mayor **series de tiempo** * 578B / 1024^2.

La Figura 5 muestra la topología del sistema, las comunicaciones y la correspondencia entre los elementos ejecutables y los nodos de proceso:

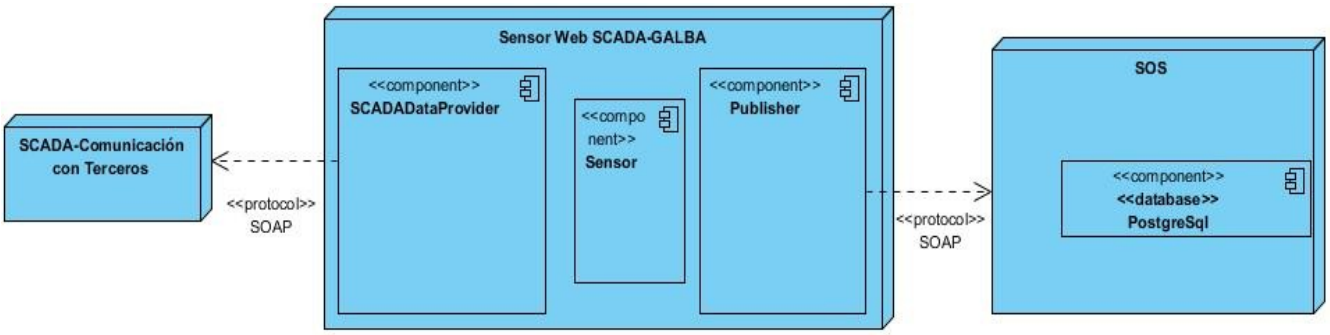


Figura 5: Despliegue de la solución.

Los componentes que integran el despliegue son explicados a continuación:

- **SCADA-Comunicación con Terceros** es un servicio de SCADA-GALBA el cual provee información del mismo a través del protocolo SOAP.
- **SOS** es el servidor que provee las observaciones de los sensores a los clientes de sensores, este incluye una base de datos postgresSql que usa para almacenar sus datos.
- **Sensor web SCADA-GALBA** compuesto a su vez por SCADA-DataProvider que se comunica a través del protocolo SOAP con el servicio SCADA-Comunicación con Terceros para extraer los datos expuestos por SCADA-GALBA, los cuales serán codificado por el componente Sensor y serán enviados al SOS a través del protocolo SOAP.

3.2 Pruebas

Unos de los principales objetivos de AUP es el uso de las pruebas para verificar el buen funcionamiento del código que se desarrolla. Esto permite aumentar la calidad del sistema reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y la detección del mismo.

Las pruebas son divididas en tres grupos: pruebas unitarias, de cobertura y de aceptación. Las pruebas unitarias son desarrolladas por los programadores y se encargan de verificar el código automáticamente, las pruebas de aceptación son para verificar que cada historia de

usuario llegó al final de cada iteración cumpliendo todas las funcionalidades previstas y las pruebas de cobertura son para conocer la medida de la cantidad de código que se ejecuta, es válida, tanto para aplicarse en una aplicación como a las propias pruebas unitarias.

	observationid [PK] bigint	value numeric(19,2)
1	2	45.00
2	5	45.00
3	8	45.00
4	11	45.00
5	14	45.00
6	17	45.00
7	20	45.00

Figura 6: Resultado de la prueba de aceptación de la HU 4

3.2.1 Pruebas unitarias

Las pruebas unitarias o pruebas de caja blanca son también llamadas pruebas modulares ya que permiten determinar si una unidad, módulo o clase del programa está listo y correctamente terminado. Es recomendado separar los módulos de acuerdo a su funcionalidad, si son muy grandes y contienen muchas funcionalidades, estos serán muy complejos de probar y al encontrar un error será difícil de ubicar la funcionalidad defectuosa y corregirla [26].

Para hacer empleo de estas pruebas se siguió el procedimiento conocido como desarrollo dirigido por pruebas (*Test Drived Development*, por sus siglas en inglés TDD), el cual es una práctica de programación que involucra escribir las pruebas primero y refactorización de código.

Se comienza escribiendo las pruebas esperando que estas fallen, luego se refactoriza el código y se vuelve a probar hasta lograr que las pruebas pasen satisfactoriamente. Este procedimiento hace que los requerimientos sean traducidos a pruebas, de modo que el hecho

de que una prueba pase satisfactoriamente apunta a que se ha cumplido con el requerimiento correspondiente a esta.

Las pruebas unitarias se realizaron en la consola con apoyo de Pytest. Fueron ejecutadas cinco veces hasta lograr que no existieran no conformidades, asegurando así el correcto funcionamiento de cada uno de las clases por independiente. A continuación se muestran las cinco iteraciones:

Iteración 1: Como suele ocurrir en TDD las pruebas fueron implementadas con anterioridad, se obtuvo 13 no conformidades.

Figura 7: Prueba Unitaria Iteración 1.

Iteración 2: Se obtuvo 5 no conformidades.

```
============================= test session starts ==============================
platform linux2 -- Python 2.7.8 -- py-1.4.26 -- pytest-2.6.4 -- /home/leiserfg/.virtualenvs/tester/bin/python2
plugins: cov, pep257
collected 13 items

test/test_sensor.py::test_coords PASSED
test/test_sensor.py::test_insert_soap FAILED
test/test_sensor.py::test_observation FAILED
test/test_types.py::test_count_sml PASSED
test/test_types.py::test_category_sml PASSED
test/test_types.py::test_data_array_sml FAILED
test/test_types.py::test_data_record_sml FAILED
test/test_types.py::test_data_array_om FAILED
test/test_utils.py::test_no_scheme_url PASSED
test/test_utils.py::test_bad_scheme_url PASSED
test/test_utils.py::test_emptypath_url PASSED
test/test_utils.py::test_ok_url PASSED
test/test_utils.py::test_gen_urls PASSED

=================================== FAILURES ===================================
_____________________________ test_insert_soap _____________________________
```

Figura 8: Prueba Unitaria Iteración 2.

Iteración 3: Se obtuvieron 2 no conformidades.

```
platform linux2 -- Python 2.7.8 -- py-1.4.26 -- pytest-2.6.4 -- /home/leiserfg/.virtualenvs/tester/bin/python2
plugins: cov, pep257
collected 13 items

test/test_sensor.py::test_coords PASSED
test/test_sensor.py::test_insert_soap FAILED
test/test_sensor.py::test_observation FAILED
test/test_types.py::test_count_sml PASSED
test/test_types.py::test_category_sml PASSED
test/test_types.py::test_data_array_sml PASSED
test/test_types.py::test_data_record_sml PASSED
test/test_types.py::test_data_array_om PASSED
test/test_utils.py::test_no_scheme_url PASSED
test/test_utils.py::test_bad_scheme_url PASSED
test/test_utils.py::test_emptypath_url PASSED
test/test_utils.py::test_ok_url PASSED
test/test_utils.py::test_gen_urls PASSED

=================================== FAILURES ===================================
_____________________________ test_insert_soap _____________________________

    def test_insert_soap():
```

Figura 9: Prueba Unitaria Iteración 3.

Iteración 4: Se obtuvo 1 no conformidad.

```
platform linux2 -- Python 2.7.8 -- py-1.4.26 -- pytest-2.6.4 -- /home/leiserfg/.virtualenvs/tester/bin/python2
plugins: cov, pep257
collected 13 items

test/test_sensor.py::test_coords PASSED
test/test_sensor.py::test_insert_soap PASSED
test/test_sensor.py::test_observation FAILED
test/test_types.py::test_count_sml PASSED
test/test_types.py::test_category_sml PASSED
test/test_types.py::test_data_array_sml PASSED
test/test_types.py::test_data_record_sml PASSED
test/test_types.py::test_data_array_om PASSED
test/test_utils.py::test_no_scheme_url PASSED
test/test_utils.py::test_bad_scheme_url PASSED
test/test_utils.py::test_emptypath_url PASSED
test/test_utils.py::test_ok_url PASSED
test/test_utils.py::test_gen_urls PASSED

=================================== FAILURES ===================================
_____________________________ test_observation _____________________________

    def test_observation():
```

Figura 10: Prueba Unitaria Iteración 4.

Iteración 5: Ultima iteración de las pruebas. Asegurando el correcto funcionamiento de todas las funcionalidades de la biblioteca.

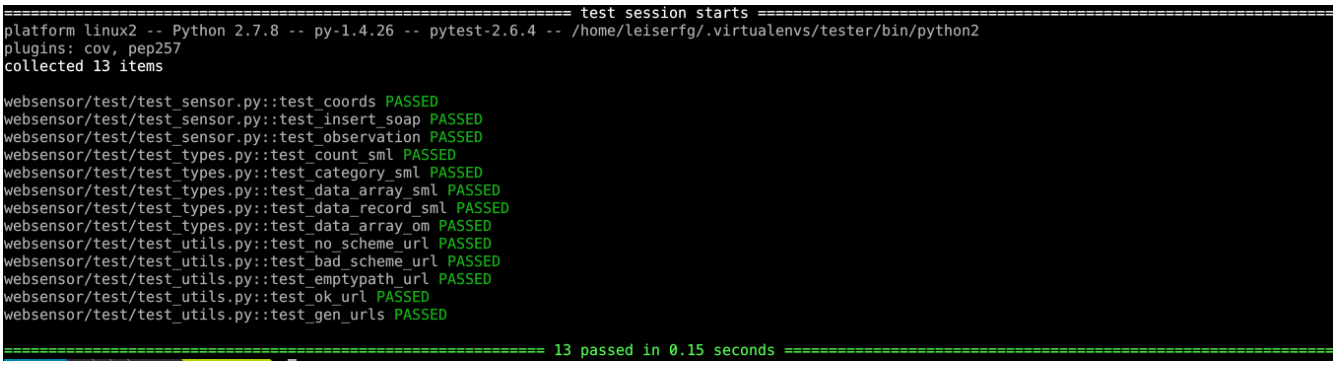


Figura 11: Prueba Unitaria Iteración 5.

Como se mencionó anteriormente, estos resultados implican que todo el código probado funciona correctamente.

3.2.2 Pruebas de cobertura.

Las pruebas de cobertura reportan la cantidad de sentencias ejecutables que son encontradas tras analizar un programa determinado. En ellas las declaraciones que generan código ejecutable también son tomadas como sentencias ejecutables, también las sentencias de control de flujo como *if*, *for* y *switch* son tratadas como tal; no así las sentencias implícitas como los *return* omitidos.

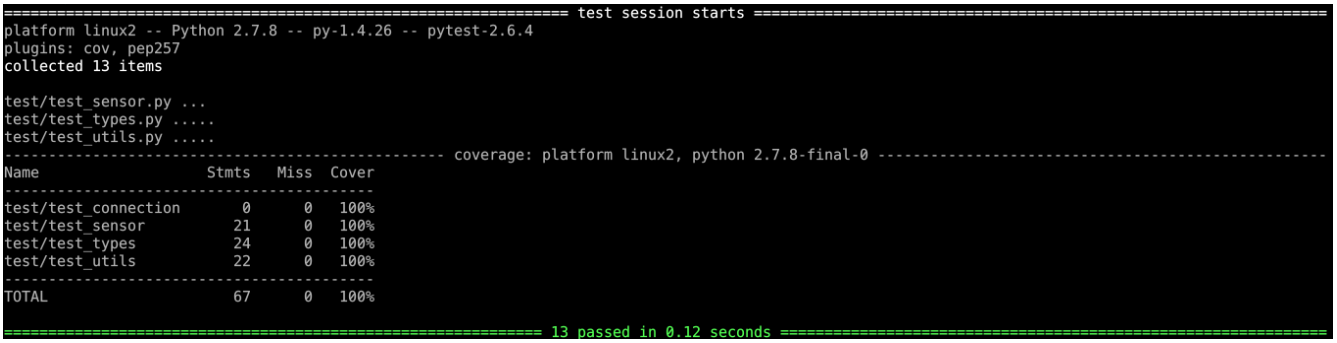


Figura 12: Prueba de Cobertura.

Con el uso de la consola y con el apoyo de Pytest y su extensión pytest-cov, se realizó la pruebas de cobertura. Esta fue basada en las pruebas unitarias, arrojando como resultado que

el 100% de las sentencias eran ejecutadas, por lo que se afirma que las pruebas unitarias abarcan todas las funcionalidades para las cuales fueron creadas a comprobar.

3.2.3 Pruebas de aceptación

El uso de cualquier producto debe estar justificado por las ventajas que ofrece. Sin embargo antes de empezar a usarlo es muy difícil determinar si sus ventajas realmente justifican su uso. Para esta determinación una herramienta que puede emplearse es la conocida como "Pruebas de aceptación". En estas pruebas se evalúa el grado de calidad del software con relación a todos los aspectos para que el uso del producto se justifique. Para eliminar la influencia de conflictos de intereses, estas pruebas no son responsabilidades de los desarrolladores. Para la preparación, ejecución y la evaluación de las pruebas de aceptación ni siquiera hacen falta conocimientos informáticos; solo son necesarios conocimientos de métodos y técnicas de pruebas. Las personas adecuadas o el equipo adecuado para llevar a cabo la prueba de aceptación disponen de estos conocimientos, llevando a cabo a partir de las HU las pruebas de aceptación en cada iteración. Especificando uno o varios escenarios para verificar que la HU ha sido correctamente implementada [27].

Caso de Prueba de Aceptación.	
Código: HU_1_P1	**Historia de Usuario:** 1
Nombre: Verificar la recolección de los datos.	
Descripción: Es necesario comprobar la correcta recolección de los datos del SCADA-GALBA.	
Condiciones de ejecución: El servicio de terceros del Servidor de SCADA-GALBA debe ser accesible desde la máquina donde se van a hacer las pruebas.	
Resultado esperado: Se mostraran los puntos que se están observando.	
Evaluación de la prueba: Bien.	

Tabla 23: Caso de Prueba de Aceptación. Historia de Usuario 1.

La HU_1_P1 se realizó con el uso de la consola, ejecutando el *__init__.py* del módulo del sensor de SCADA-GALBA en el modo verboso pasándole la configuración del sensor. A continuación se muestra el resultado de la dicha prueba de aceptación donde se puede

comprobar que corresponde con el resultado esperado.

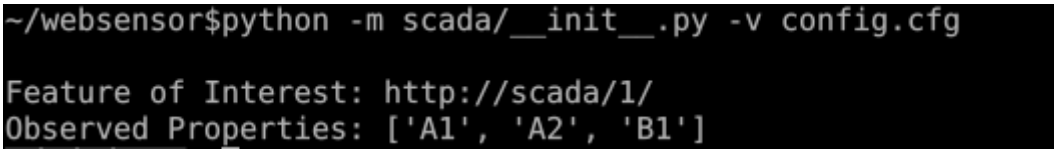


Figura 13: Resultado de la prueba de aceptación de la HU 1

Caso de Prueba de Aceptación.	
Código: HU_3_P1	**Historia de Usuario:** 3
Nombre: Verificar la inscripción al SOS.	
Descripción: Se verifica la inscripción del sensor en el SOS	
Condiciones de ejecución: El SOS debe ser accesible desde la máquina donde se van a hacer las pruebas.	
Resultado esperado: El sensor debe estar registrado en la tabla donde se guarda la instancia de un sensor de la base de datos que usa el SOS. Verificando el valor http://scada/1 en la variable *feature of interest*, la cual representa l'última columna de la tabla.	
Evaluación de la prueba: Bien.	

Tabla 24: Caso de Prueba de Aceptación. Historia de Usuario 3.

La HU_3_P1 se realizó manualmente verificando que el sensor estuviera registrado en la tabla *featureofinterest* de la base de datos que usa el SOS para almacenar las propiedades de interés de cada sensor. A continuación se muestra el resultado de dicha prueba de aceptación donde se puede comprobar que corresponde con el resultado esperado.

	hibernatediscriminator character(1)	featureofinteresttypeid bigint	identifier character varying(255)
3	T	1	http://what.com/5
4	T	1	http://whate.com/5
5	T	1	http://what.com/6
6	T	1	http://what.com/8
7	T	1	http://scada/1
8	T	1	http://what.com/9

Figura 14: Resultado de la prueba de aceptación de la HU 3

Caso de Prueba de Aceptación.	
Código: HU_4_P1	**Historia de Usuario:** 4
Nombre: Verificar recepción de los datos recolectados del SCADA-GALBA en el SOS.	
Descripción: Comprobar que los datos están guardados en la tabla *countvalue* de la base de datos que usa el SOS. Dicha tabla almacena todos los valores de tipo *Count* enviados por los sensores.	
Condiciones de ejecución: El Sensor debe estar en ejecución para verificar las salidas que emite, las cuales serán mostradas en consola. El SOS debe ser accesible desde la máquina donde se van a hacer las pruebas.	
Resultado esperado: Los valores enviados deben estar registrado en la tabla *countvalue* de la base de datos que usa el SOS.	
Evaluación de la prueba: Bien.	

Tabla 25: Caso de Prueba de Aceptación. Historia de Usuario 4.

La HU_4_P1 se realizó manualmente verificando que los datos de tipo *Count* enviados por el sensor se encontraran registrados en la tabla *countvalue*. A continuación se muestra el resultado de dicha prueba de aceptación donde se puede comprobar que corresponde con el resultado esperado.

Conclusiones parciales

En el presente capítulo se describió la implementación de la biblioteca para la habilitación de Sensores Web en el SCADA-GALBA. Los resultados satisfactorios de las pruebas unitarias, de cobertura y de aceptación muestran que la misma permitirá la integración de SCADA-GALBA a redes de sensores web.

Conclusiones

- El diagnóstico inicial apoyado en la aplicación de los métodos científicos permitió identificar las funcionalidades, tecnologías, y herramientas utilizadas en el desarrollo de la solución propuesta.
- El desarrollo de una biblioteca que haga posible la habilitación de Sensores Web a partir de datos obtenidos de SCADA-GALBA, permitirá la integración de este a redes de sensores.
- Después de realizadas las pruebas unitarias, de cobertura y de aceptación se puede concluir que la solución desarrollada cumple con todos los requerimientos acordados con el cliente.

Recomendaciones

- Extender el proveedor de datos de SCADA-GALBA, de modo que no solo exponga valores de los puntos, sino que también los de otras fuentes, como el estado de las alarmas.
- Añadir a la biblioteca otras formas de encapsular SensorML y O&M para el envió de información al SOS además de SOAP, tales como POX y JSON, que son implementados por 52°North.

Referencia Bibliográfica

[1]Botts, M., Percivall, G., Reed, C. & Davidson, J. 2008. OGCtextregistered sensor web enablement: Overview and high level architecture. In: (ed.) GeoSensor networks. Springer, . 175-190pp.

[2]Simonis, I."OGC sensor web enablement architecture".Open Geospatial Consortium. ed.2008,vol ,núm. ,p.

[3]Geraci, A. and Katki, F. and McMonegal, L. and Meyer, B. and Lane, J. and Wilson, P. and Radatz, J. and Yee, M. and Porteous, H. and Springsteel, F. IEEE standard computer dictionary: Compilation of IEEE standard computer glossaries. . ed. : IEEE Press, 1991. p. . . ISBN:.

[4]Bermudez, L. and Delory, E. and O'Reilly, T. and del Rio Fernandez, J.Ocean observing systems demystified. ed.:,2009. p. 1--7.ISBN: .

[5]Simonis, I."OGC sensor web enablement architecture".Open Geospatial Consortium. ed.2008,vol ,núm. ,p. 14

[6]Simonis, I."OGC sensor web enablement architecture".Open Geospatial Consortium. ed.2008,vol ,núm. ,p. 15

[7]GIMÉNEZ SALAZAR, PABLO. Simulador web de redes de sensores para la automatización industrial. . . :, 2013.

[8]Chung, L.-K. and Fang, Y.-M. and Chang, Y.-H. and Chou, T.-Y. and Lee, B. J. and Yin, H.-Y. and Baranski, B.A SOA based debris flow monitoring system. ed.:,2009. p. 1-6.ISBN: .

[9]Stasch, C. and Walkowski, A. C. and Jirka, S."A geosensor network architecture for disaster management based on open standards".Digital Earth Summit on Geoinformatics. ed.2008,vol ,núm. ,p. 54-59

[10]Schimak, G. and Havlik, D."Sensors anywhere sensor web enablement in risk management applications".Ercim News. ed.2009,vol 76,núm. ,p. 40-41

[11]Jirka, S. and Bröring, A. and Stasch, C.Applying OGC Sensor Web Enablement to risk monitoring and disaster management. ed.:,2009. p. .ISBN: .

[12]Raape, U. and Tessmann, S. and Wytzisk, A. and Steinmetz, T. and Wnuk, M. and Hunold, M. and Strobl, C. and Stasch, C. and Walkowski, A. C. and Meyer, O. and others."Decision support for tsunami early warning in indonesia: The role of standards.". ed.2009,vol ,núm. ,p.

[13]Cox, S."Observations and Measurements-Part 1-Observation schema (OpenGIS Implementation Standard OGC 07-022r1)".Open Geospatial Consortium Inc., Tech. Rep. ed.2007,vol 8,núm. ,p.

[14]Simonis, I."OGC sensor web enablement architecture".Open Geospatial Consortium. ed.2008,vol ,núm. ,p. 23-24

[15]Simonis, I."OGC sensor web enablement architecture".Open Geospatial Consortium. ed.2008,vol ,núm. ,p. 28

[16]Simonis, I."OGC sensor web enablement architecture".Open Geospatial Consortium. ed.2008,vol ,núm. ,p. 32

[17]della Terra, I.S. 2014. istSOS project. https://geoservice.ist.supsi.ch/projects/istsos/index.php/Welcome_to_istSOS_project. Accessed 12

[18]52north . 52N Sensor Web - About. http://52north.org/about. Accessed

[19]Sánchez, T. R."Metodología de desarrollo para la Actividad productiva de la UCI". ed.2014,vol ,núm. ,p.

[20]Swaroop, C."A Byte of Python".Enlace web. ed.2003,vol ,núm. ,p.

[21]pytest's Team . pytest: helps you write better programs. http://pytest.org/latest/. Accessed

[22]Apache. Apache Subversion. . ed. : , . p. . . ISBN: .

[23]Garzás, J. . Los patrones GRASP. http://www.javiergarzas.com/2014/08/los-patrones-grasp.html. Accessed

[24]52north . 52N Sensor Web Community - Sensor Observation Service. http://52north.org/communities/sensorweb/sos/index.html. Accessed

[25]52north . Sensor Observation Service IV Documentation. https://wiki.52north.org/bin/view/SensorWeb/SensorObservationServiceIVDocumentation. Accessed

[26]Oré, A. . Pruebas Unitarias.
http://www.calidadysoftware.com/testing/pruebas_unitarias1.php. Accessed
[27]pruebasdesoftware . La prueba de aceptación.
http://pruebasdesoftware.com/pruebadeaceptacion.htm. Accessed

Bibliografía

Cox, B. J."Object oriented programming". ed.1985,vol ,núm. ,p.

Zhu, H. and Hall, P. A. and May, J. H."Software unit test coverage and adequacy".Acm computing surveys (csur). ed.1997,vol 29,núm. 4,p. 366--427

Hamill, P. Unit Test Frameworks: Tools for High-Quality Software Development. . ed. : " O'Reilly Media, Inc.", 2004. p. . . ISBN: .

Meyer, B."Object oriented software construction". ed.1988,vol ,núm. ,p.

Wirfs-Brock, R. and Wilkerson, B. and Wiener, L."Designing object-oriented software". ed.1990,vol ,núm. ,p.

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. & Berners-Lee, T. 1999. Hypertext transfer protocol--HTTP/1.1. . , .

Sanner, M. F. and others."Python: a programming language for software integration and development".J Mol Graph Model. ed.1999,vol 17,núm. 1,p. 57--61

Curbera, F. and Duftler, M. and Khalaf, R. and Nagy, W. and Mukhi, N. and Weerawarana, S."Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI".IEEE Internet computing. ed.2002,vol ,núm. 2,p. 86--93

Anexos

1. Estándar de codificación Pocoo

El estándar de codificación Pocoo está basado en el la guía de estilo oficial de python recogida en la PEP8 con ligeras modificaciones:

- Sangrado:
  - 4 espacios reales, por ningún concepto se pueden emplear tabulaciones.
- Longitud máxima de las líneas:
  - 79 caracteres (a lo sumo 84 si es sumamente necesario).
- Para continuar sentencias largas:
  - Usa barra inversa (\) para separar las líneas y en la siguiente sangra con 4 espacios o hasta la altura del signo de igual (=) o punto (.) correspondiente.
  - En caso de que separar una línea con agrupadores (paréntesis, llaves), sangra hasta la altura del agrupador correspondiente.
  - En caso de una lista demasiado extensa, corta siempre tras el primer corchete y luego se coloca un elemento por línea.
- Líneas en blanco:
  - Las funciones y clases en el ámbito de un módulo se separan con dos líneas en blanco.
- Reglas generales de espaciado:
  - No separar con espacio los operadores unarios ni el lado interno de los paréntesis.
  - Separar con espacio entre operadores binarios.

- Comparaciones:
  - Nunca comprar valor con variable (conocida como sentencias Yoda), siempre variable con valor.
  - Con tipos arbitrarios usar == y !==
  - Con singletons usar is y not is.
  - Nunca comparar con True o False, usa el valor de la expresión
  - Para comprobar la inexistencia de un valor en un contenedor emplear el operador *not in*.
- Nombrado:
  - Nombrar las clases con CamelCase manteniendo las mayúsculas en los acrónimos.
  - Las variables con minúsculas separando las palabras con guion bajo.
  - Las constantes con mayúsculas separando las palabras con guion bajo.
  - Los miembros protegidos comienzan con un guion bajo, los mixins con dos.
  - El primer parámetro de un método de clase será siempre cls.
  - El primer parámetro de un método de instancia será siempre self.
- DocStrings (cadenas de documentación propias de la sintaxis de python).
  - Todas serán en reStructuredText[8].
- Comentarios
  - Serán en reStructuredText, en caso de ser sobre un atributo comenzarán con dos puntos (:).

8 Lenguaje de marcado empleado comúnmente para documentar en python.

2. Tarjetas CRC

Tarjeta CRC	
Clase: Count	
Superclase: _BaseType	
Subclase: -	
Responsabilidad	**Colaboración**
1. Serializa los valores enteros en el estándar SensorML. 2. Serializa los valores enteros en el estándar O&M	Módulo: XML

Tabla 26: Tarjeta CRC clase Count

Tarjeta CRC	
Clase: Boolean	
Superclase: _BaseType	
Subclase: -	
Responsabilidad	**Colaboración**
1. Serializa los valores de tipo boolean en el estándar SensorML. 2. Serializa los valores enteros en el estándar O&M	Módulo: XML

Tabla 27: Tarjeta CRC clase Boolean

Tarjeta CRC	
Clase: Category	
Superclase: _BaseType	
Subclase: -	
Responsabilidad	**Colaboración**
1. Serializa el valor que puede ser escogido entre un conjunto de valores en el estándar SensorML. 2. Serializa el valor que puede ser escogido entre un conjunto de valores en el estándar O&M.	Módulo: XML

Tabla 28: Tarjeta CRC clase Category

Tarjeta CRC	
Clase: Quantity	
Superclase: _BaseType	
Subclase: -	
Responsabilidad	**Colaboración**
1. Serializa los valores reales y/o definiendo la unidad de medida en el estándar SensorML. 2. Serializa los valores reales y/o definiendo la unidad de medida en el estándar O&M.	Módulo: XML

Tabla 29: Tarjeta CRC clase Quantity

Tarjeta CRC	
Clase: DataArray	
Superclase: _BaseType	
Subclase: -	
Responsabilidad	**Colaboración**
3. Serializa los valores de un arreglo en el estándar SensorML. 4. Serializa los valores de un arreglo en el estándar O&M.	Módulo: XML

Tabla 30: Tarjeta CRC clase DataArray

Tarjeta CRC	
Clase: DataRecord	
Superclase: _BaseType	
Subclase: -	
Responsabilidad	**Colaboración**
1. Serializa los tipos de datos como diccionarios o mapas en el estándar SensorML. 2. Serializa los tipos de datos como diccionarios o mapas en el estándar O&M.	Módulo: XML

Tabla 31: Tarjeta CRC clase DataRecord