Universidad Nacional de La Plata

Facultad de Informática

Plataformas para la creación de *Mashups* Sensibles al Contexto en Entornos de Inteligencia Ambiental

Trabajo Final Para la Obtención de Título de Especialista en Ingeniería de Software

Autor: Diego Alberto Godoy

Director: Dr. Eduardo O. Sosa

Co-Director: Dr. Gustavo Rossi

2013

Contenido

Figuras

Nomenclatura, Abreviaturas y Acrónimos

AJAX	Asynchronous JavaScript And XML (JavaScript asíncrono y XML
AmI	Ambient Intelligence - Inteligencia Ambiental-
API	Application Programming Interface - Interfaz de Programación de Aplicaciones
ATOM	Es un protocolo simple basado en HTTP para crear o actualizar recursos en Web.
Ubicua/o	Del lat. ubīque, en todas partes. Que está presente a un mismo tiempo en todas partes.
Contexto	Conjunto de información necesaria para adaptar el comportamiento o la funcionalidad de un sistema.
CRM	Customer Relationship Management.
ERP	Enterprise Resource Planning- Planificación de Recursos Empresariales
Folksonomía	Es una indexación social por medio de etiquetas simples en un espacio de nombres llano, sin jerarquías ni relaciones de parentesco predeterminadas.
GPS	Global Positioning System - Sistema de Posicionamiento Global
GUI	Graphical User Interface - Interface Gráfica de Usuario.
HCI	Human-Computer Interaction - Interacción Hombre/ Maquina
ISTAG	Information Society Technologies Program Advisory Group
JSON	Java Script Object Notation - Lenguaje de Notación de Objetos Java Script
Mashup	Aplicación basada en Web que se crea mediante la combinación y procesamiento de recursos en línea de terceros.
Ontología	formulación de un exhaustivo y riguroso esquema conceptual dentro de uno o varios dominios

OWL	Web Ontology Language, -Language de Ontología Web
Plataformas de Software	Las plataformas de software pueden ser un sistema operativo, un entorno de programación, o (más comúnmente) una combinación de ambos.
POI	Point of Interest - Punto de Interés
PULL	Estrategia donde el usuario "pregunta" al servidor por una nueva información.
PUSH	Estrategia describe un estilo de comunicaciones donde la petición de una transacción se origina en el servidor y llega automáticamente al usuario.
RDF	Resource Description Framework - Marco de Descripción de Recurso
REST	Representational State Transfer . - Transferencia de Estado Representacional.
RIA	Rich Internet applications - Aplicaciones de Internet Enrriquecidas.
RSS	Really Simple Syndication - Sindicación Realmente Simple
Screen scraping	Técnica de programación que consiste en tomar una presentación de una información para, mediante ingeniería inversa, extraer los datos que dieron lugar a esa presentación.
Sensible al Contexto	Se refiere al uso del contexto para automatizar un sistema de software, modificar su interface, su comportamiento y proveer la máxima flexibilidad en términos de servicios.
SGML	Standard Generalized Markup Language - Lenguaje de Marcado Estándar Generalizado.
SOAP	Simple Object Access Protocol - Protocolo de Acceso simple a Objetos.
Tag	Etiquetas de identificación.
UML	Unified Modeling Language. Lenguaje de Modelado Unificado.
URI.	Uniform Resource Identifier - Identificador Uniforme de Recursos

W3C	Comunidad internacional para desarrollar estándares Web.
Web 2.0	Evolución de la Web original. La web 2.0 está basada en servicios, en la cual los clientes colaboran y comparten información en línea, aportando así una nueva forma de interacción social
Web semántica	La Web Semántica es una extensión de la Web actual dotada de significado, esto es, un espacio donde la información tendría un significado bien definido, de manera que pudiera ser interpretada tanto por agentes humanos como por agentes computarizados.
Web service	Es una aplicación que puede ser descripta, publicada, localizada e invocada a través de una red, generalmente Internet.
Widgets	Widget es una abreviación de las palabras window y gadget.
Wrappers	Son envolturas que se utilizan para abstraer la complejidad o diferencia en la forma de utilizar un determinado Servicio.
WSDL	Web Services Description Language -. Leguage de descripción de servicio web.
WSN	Redes de Sensores Inalámbricos .
XML.	Extensible Markup Language - Lenguaje de Marcado Extensible.

Introducción

En los últimos años hemos sido testigos del importante desarrollo alcanzado por dos tecnologías: por un lado las redes de sensores inalámbricos WSN (Redes de Sensores Inalámbricos por sus siglas en inglés) y los dispositivos móviles y por otro la Web.

En cuanto a la primera de estas tecnologías, se han desarrollado grandes avances en redes inalámbricas como las WSN, como ser: protocolos más eficientes, mayores anchos de banda, mayores áreas de cobertura, nuevos tipos de sensores, entre otros. Además los dispositivos móviles cuentan cada vez más, con mayor poder de cómputo, pantallas más grandes y se consiguen a un menor precio. Estos avances facilitan el desarrollo de aplicaciones en donde la movilidad y ubicuidad son parte esencial.

Sin duda una de las características más importantes de los sistemas móviles, es que brindan la posibilidad de aprovechar la posición geográfica para asistir al usuario mientras este se desplaza. Ejemplos de ello son los sistemas que guían a los usuarios en las visitas a museos, sistemas de navegación basados en GPS, o proyectos como Friend Finder de A&T que notifica la presencia de personas geográficamente cercanas.

Hoy en día el desarrollo de sistemas para escenarios móviles ha aumentado considerablemente. Se ha pasado desde aplicaciones casi experimentales o utilizadas solo en ambientes académicos, hasta algunas aplicaciones comerciales como las orientadas a negocios, el uso personal o el ocio.

De la misma manera que las redes inalámbricas, el aumento del poder de cómputo y capacidad de sensado de los dispositivos móviles supone una revolución, la web también fue teniendo grandes avances en cuanto a la forma de presentar la información a los usuarios. Estos avances dieron origen a una evolución de la web, llamada web 2.0, concepto desarrollado por Tim O'Reilly, en donde los usuarios ya no son únicamente consumidores de información si no que también son capaces de producirla, y donde la experiencia del usuario al

usar estos sistemas ganó en ergonomía con las mejoras de las interfaces gráficas, posibilitando así personalizar las aplicaciones y adaptarlas a las necesidades concretas de los usuarios. En esta nueva web ya no son suficientes sitios donde solo puedan obtener datos de una única fuente, sino, que se hace sumamente necesario contar con recursos de diversos orígenes integrados de forma sinérgica. Para ello se han desarrollado plataformas que permiten hacer mezclas o *mashups* tanto de información como de comportamiento de las aplicaciones disponibles en la red. La particularidad de estas mezclas es que son los mismos usuarios los que tienen la posibilidad de crearlas para cumplir con sus requisitos y no necesitan de un programador o conocer lenguajes como Java o C++. Un ejemplo de estas herramientas es *Yahoo Pipes*.

La AmI no está ajena a los avances de estas dos tecnologías (los dispositivos móviles y la web 2.0) y se ve beneficiada por ambas. Estos avances permiten que la AmI se acerque cada vez más al cumplimento de la visión del Information Society Technologies Program Advisory Group (ISTAG) de la Unión Europea en la cual "…personas rodeadas de interfaces inteligentes embebidas en objetos de la vida cotidiana, permitan la interacción de forma natural y sin esfuerzo con diferentes sistemas de información...". Es necesario destacar que una de las características fundamentales de las aplicaciones basadas en AmI es la sensibilidad al contexto. Según la definición de Dey, "una aplicación es sensible al contexto si esta hace uso de información del contexto, para dar nueva información o servicios relevantes al usuario dependiendo de actividad actual del usuario", lo cual es perfectamente compatible con la visión de la AmI.

Objetivos del Trabajo

General:

Analizar las plataformas existentes para la creación de *mashups* Sensibles al Contexto en Entornos de AmI.

Específicos:

- Identificar los requisitos de las aplicaciones basadas en AmI.
- Caracterizar las aplicaciones Sensibles al Contexto.

- Estudiar los requisitos y las herramientas para la realización de mashups.
- Describir las Plataformas existentes para creación de Mashups Sensibles al contexto en entornos de AmI.
- Proponer una discusión acerca de la temática que de lineamientos para trabajos futuros.

Organización del Trabajo

En el Capítulo 1 se verá la definición de formal de AmI, su visión, los requisitos básicos de tecnología, los requisitos para el desarrollo de aplicaciones y servicios, y además algunas aplicaciones prácticas existentes. En el Capítulo 2 se estudiarán las aplicaciones sensibles al contexto como un componente más de la AmI, se verán distintas definiciones de contexto, una taxonomía de elementos de contexto y formas de representación de información contextual. En el Capítulo 3 se trataran los *mashups* dando su definición, arquitectura básica, técnicas y tecnologías de soporte como así también herramientas existentes en el mercado para su creación. En el Capítulo 4 se presentan tres plataformas/arquitecturas incipientes para el desarrollo de *mashups* sensibles al contexto. Por último en el capítulo 5 se presenta la discusión del presente trabajo.

1. Inteligencia Ambiental

En los últimos años hemos sido testigos de la evolución y creciente miniaturización de los dispositivos de cómputo, que hace posible que pequeños procesadores y diminutos sensores se integren cada vez más en objetos cotidianos. Esto lleva a la desaparición de los dispositivos de entrada y salida de una computadora personal estándar, tales como teclados, *mouses* y pantallas. Además, también se han desarrollado grandes avances en redes inalámbricas, protocolos más eficientes, mayores anchos de banda, mayores áreas de cobertura, entre otros avances. Estos Avances facilitan el desarrollo de aplicaciones en donde la movilidad y ubicuidad son parte esencial.

Mark Weiser había previsto ésta evolución hace casi 20 años y la publicó en su artículo "The Computer for the 21st Century" [1]. En un trabajo posterior Weiser se planteaba los siguientes interrogantes “¿Cuál es la idea de la computadora del futuro? ¿El agente inteligente? , ¿La televisión (multimedia)?, ¿El mundo gráfico 3-D (realidad virtual)? ¿El equipo de voz ubicua (la palabra esta escrita de diferentes maneras) de la película StarTrek?, ¿Las interfaces gráficas de usuarios –GUI-, elaboradas y seleccionadas?, ¿La máquina que por arte de magia cumpla y acceda a nuestros deseos?” [2]. La respuesta que él mismo ha dado es: ninguna de las anteriores. La sencilla razón es que todos estos conceptos comparten un defecto básico: se llevan a la práctica con algo tangible, visible.

Es así que Weiser definió el concepto de "computación ubicua", refiriéndose a las computadoras omnipresentes al servicio de las personas en su vida cotidiana, en el hogar y en el trabajo, y funcionando de manera invisible y no intrusiva. Los principios enunciados originalmente por Weiser han sido validados por diferentes investigadores e ingenieros, convirtiéndose en un escenario real en ciertos ámbitos y extendiéndose a escala global [3].

En 1999 ISTAG ha utilizado el término "AmI[1]" de manera similar para describir una visión donde "las personas estarán rodeadas de interfaces inteligentes e intuitivas embebidas en objetos cotidianos de nuestro alrededor y un entorno que reconocen y responden a la presencia de individuos de manera invisible" [4]. Es así que el objetivo de la AmI consiste en la creación de espacios donde los usuarios interaccionen de forma natural y sin esfuerzo con los diferentes sistemas, gracias a que las tecnologías de computación y comunicación convierten a estos entornos en invisibles para el usuario al estar siempre presentes e integradas en los objetos cotidianos del mismo. De esta forma, es la propia tecnología la que se adapta a los individuos y a su contexto, actuando de forma autónoma, facilitándoles la realización de sus tareas diarias y la comunicación entre ellos y con el entorno.

Los avances alcanzados en la tecnología de las redes inalámbricas en general, nos han colocado en la puerta de una nueva era, en la cual pequeños equipos inalámbricos nos proveerán acceso a la información en cualquier momento y en cualquier lugar, permitiéndonos asimismo participar activamente en la creación de "espacios inteligentes".

La visión de un futuro lleno de de objetos inteligentes y de interacción cotidiana ofrece una amplia gama de posibilidades en la construcción de sistemas que soporten las actividades de la vida diaria de forma más eficiente.

En estos entornos sería posible [5]:

- Capturar las experiencias diarias, mediante la monitorización y recogida de toda la información asociada al entorno, referente al contexto, los usuarios y sus actividades, a lo que se conoce como "información de contexto".

- Acceder a la información, referida tanto al propio sistema, como a información nueva obtenida del exterior a través de Internet, de manera ubicua y eficiente.

[1] Del inglés "Ambient Intelligence" (Aml)

- Soportar la comunicación y colaboración. El sistema debe ofrecer capacidades de comunicación con el menor esfuerzo posible y en cualquier lugar. La implementación debe ser extensible a todos los escenarios con los que conviva el usuario, interconectando los sistemas de forma eficiente.

- Desarrollar entornos sensibles al contexto. Los entornos deben ser sensibles tanto a la información del entorno, como a la información del usuario. Para ello el sistema dispondrá de herramientas que capturen esa información, la procesen, y en función de los resultados de ese análisis, modifiquen su comportamiento.

- Proporcionar nuevas formas de interacción hombre-máquina. Mediante una amplia variedad de interfaces de usuario, este interactúa con su entorno. Estas interfaces deben ser lo más naturales, ubicuos y transparentes posible. Además, deben ser multimodales para adaptarse a la gran diversidad de entornos con los que convive el usuario y a la heterogeneidad de los dispositivos de interacción con el sistema.

- Ejercer de guía automática, es decir, el sistema es capaz de detectar a los visitantes y en función de su perfil, facilitarles información y guiarles por ese entorno.

- Facilitar el aprendizaje y entrenamiento de diversas actividades.

1.1. Requisitos de Tecnología

Para que esta visión, un tanto futurista, que pretende mejorar la calidad de vida de las personas se haga realidad es necesario progresar en una serie de tecnologías, a las que las comunidades de científicos y ingenieros contribuirán seguramente en los próximos años. Algunas de ellas se mencionan a continuación:

- Hardware miniaturizado usando nanotecnología, dispositivos inteligentes y redes de sensores inalámbricos (WSN) que capten el entorno.

- Una infraestructura de comunicación móvil y fija basada en web, donde redes de comunicaciones inalámbricas y cableadas interactúen y converjan.

- Redes de dispositivos dinámicas y masivamente distribuidas, sin necesidad de servidores centrales con capacidad de cooperación.

- Interfaces de usuario mucho más naturales y próximas al ser humano como voz o gestos.

- Sensibilidad al contexto; donde nuestra posición, identidad, o actividad sirvan como parámetros de entrada implícitos para permitir a un entorno inteligente conocer nuestra situación actual y obrar en consecuencia.

- Seguridad y tolerancia a fallos, software auto-reparable, robusto y seguro, evitando que la tecnología invada nuestra privacidad o permita que los que controlen la tecnología, como las empresas o el gobierno, se aprovechen de ello.

Por lo anterior, es claro que la AmI interrelaciona diferentes tecnologías, entre ellas la sensorial (sensores y), las redes (de dispositivos y de comunicaciones), interfaces hombre-máquina (HCI), la computación ubicua y la Inteligencia Artificial; para que en conjunto provean servicios de manera sencilla y flexible a los usuarios. En la Figura 1 se puede ver una representación de la relación de la AmI con otras aéreas de la computación.

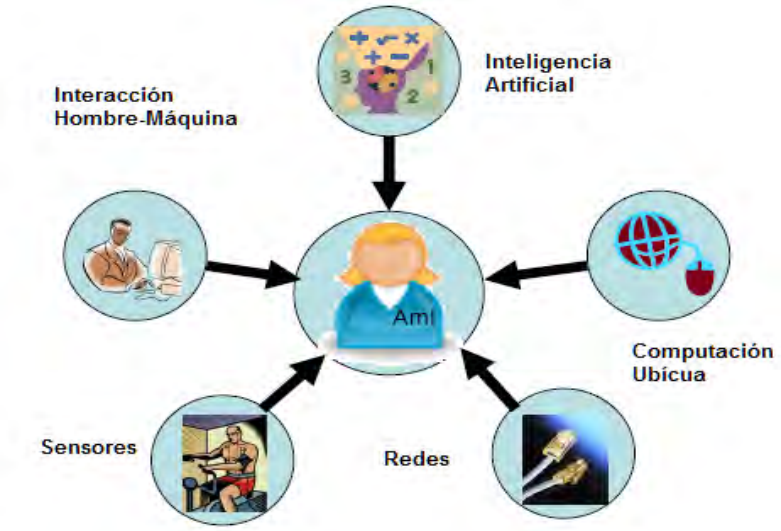


Fig. 1 - Relación de la AmI con otras áreas.

Además de las mencionadas existen otras tecnologías, particularmente de la Web, que podrían ser aplicadas para resolver algunos de los retos que se plantean en la AmI. Entre estas tecnologías se pueden mencionar la Web 2.0 y la Web semántica, las cuales se han ido desarrollando a un gran ritmo y ya existen estándares definidos que facilitarían su aplicación en la AmI.

1.2. Requisitos para aplicaciones y servicios.

Para cumplir con el objetivo de la AmI de proporcionar servicios inteligentes para el usuario, se debe tener software que cambie su comportamiento según el contexto actual anticipándose a las intenciones del usuario y que pueda a su vez adaptarse, luego que el usuario actúa y cambia el contexto. Más concretamente, se requiere de la integración del estado de la técnica de varios campos de investigación informática, como la adaptación de aplicaciones, la movilidad del código en entornos, generación automática de código y de las interfaces de usuario sensibles al contexto. Por lo tanto, tener información detallada del contexto es fundamental para poder lograr el objetivo, por ello según [6] se deben tener los siguientes requisitos básicos para modelar el contexto en aplicaciones AmI:

a. **Aplicación Adaptativa:** En un entorno dinámico y contextos cambiantes, es importante que las aplicaciones admitan cierto grado de adaptación. Por lo tanto, tener información actualizada sobre el usuario, los servicios disponibles, las plataformas utilizadas, la conectividad de red, hora, lugar y otra información que puede ser censada se deben incluir en un modelo para ayudar a que la aplicación se adapte adecuadamente.

b. **Sensibilidad a los recursos:** Como los recursos en dispositivos empotrados son a veces demasiado limitados para ejecutar ciertos servicios, la información disponible acerca de la capacidad máxima y actual de los recursos disponibles, tales como la potencia de procesamiento, memoria, tiempo de duración de la batería y el ancho de banda, debe ser considerada para la adaptación de servicios o la reubicación estos para lograr una reducción en la utilización de recursos.

c. **Los servicios móviles:** Cuando la ubicación de un usuario cambia con el tiempo, todo o partes de los servicios deben ser capaces de migrar casi instantáneamente. Por lo tanto, la información detallada sobre la ejecución de la plataforma debe permitir la migración autónoma cuando, por ejemplo, existen máquinas virtuales compatibles en dos plataformas diferentes.

d. **Servicio de descubrimiento semántico:** El descubrimiento semántico basado en información de contexto mejora los protocolos basados en clave-valor permitiendo la incorporación de criterios de búsqueda automática que son relevantes para el usuario actual o el dispositivo.

e. **Generación de código:** Al especificar el sistema operativo, controladores, librerías de software y las máquinas virtuales en un dispositivo empotrado, la generación de código se puede utilizar para generar una aplicación especializada de alto nivel de servicios especializados para ampliar la gama de dispositivos en los que estos pueden utilizados.

f. **Interfaces de usuario sensibles al contexto:** Los servicios del lado de los usuarios finales, que funcionan con las limitaciones de los dispositivos móviles necesitan interfaces de usuario que se adapten a su contexto de uso. Las Interfaces de usuario deben poder adaptarse dinámicamente si el contexto cambia con el tiempo.

Estos requisitos permiten que los servicios móviles se diseñen de una manera genérica, para que con las variaciones funcionales necesarias puedan funcionar en una serie de plataformas distintas, y también para que puedan adaptarse al contexto de otros objetos tales como los servicios y recursos disponibles en su contexto.

1.3. Aplicaciones prácticas destacadas

AmI es un concepto muy amplio que se puede aplicar a campos muy diversos. La mayoría de investigaciones actuales centran sus esfuerzos en cuatro áreas principales:

- **Hogar Digital:** El hogar digital es una vivienda que ofrece servicios a sus habitantes gracias a la interconexión de una red domótica (sensores, actuadores…), una red multimedia (TV, audio…) y una red de datos (WLAN, Internet…). Dicha red suministra servicios de automatización (iluminación, suministro de agua, alarmas, climatización, riego…). La red multimedia ofrece contenidos de información y ocio basados en imágenes y sonidos (video porteros, videoconsolas, TV…). La red de datos permite distribuir la información (archivos) entre computadoras, compartir recursos (impresoras, escáneres…) y acceder a Internet. HomeLab de Philips [7] y Oxygen [8] del MIT (Figura 2) son dos ejemplos de proyectos de investigación cuyo objetivo es establecer un hogar digital.

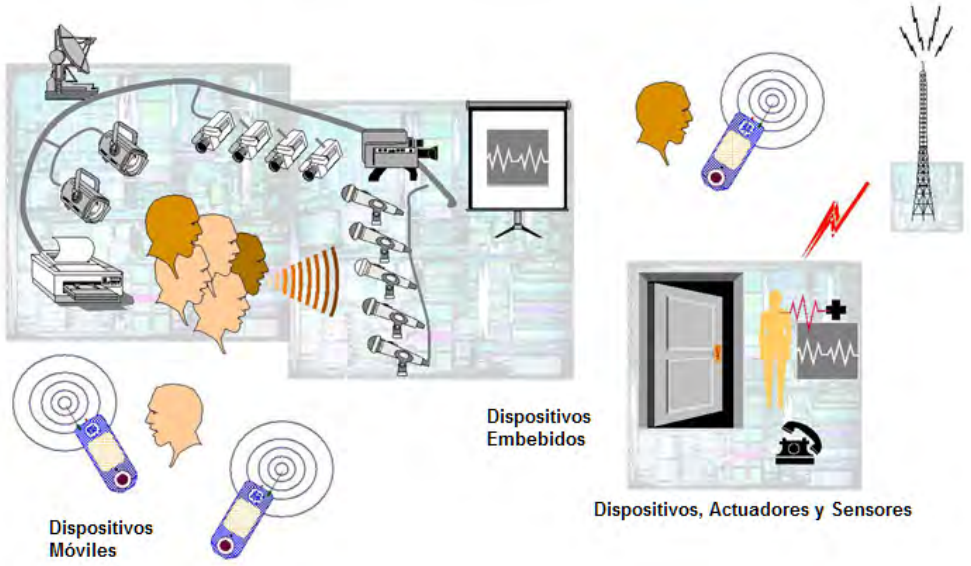


Fig. 2 - Proyecto Oxygen (*MIT Media Lab's CSAIL Group*) [8]
Reconocimiento de personas mediante captura de voz e imágenes.
Tecnología de *Smart Environment.*

- **Industria:** Estos sistemas se suelen aplicar en el proceso de producción o en el proceso de distribución de los productos. En ambos casos, estos sistemas pretenden ayudar a los trabajadores en sus tareas diarias, facilitando su trabajo al darles información precisa sobre él. Por ejemplo, el sistema AMICO [9] de Tekniker (Figura 3) puede localizar al usuario e informarle de los dispositivos que puede utilizar dependiendo de su ubicación. Además, es capaz de aprender las preferencias de cada usuario y adaptarse a ellas.

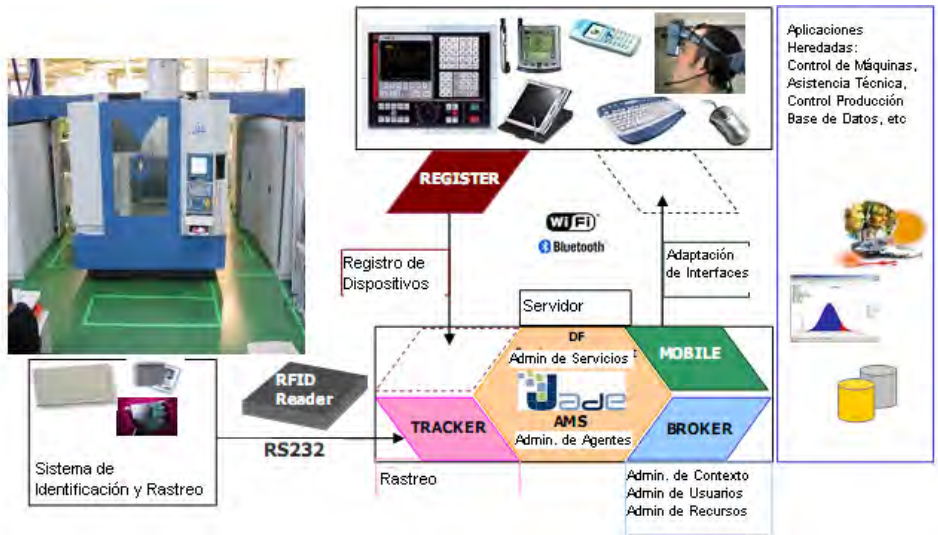


Fig. 3 – Arquitectura del sistema Amico [9]:

- **Lugares públicos:** La AmI también puede ser útil en lugares de utilización intensiva ejemplo de ello es el sistema PRISMATICA [10]: Inteligencia Ambiental en el transporte público", ha sido utilizado para realizar tareas de vigilancia trenes subterráneos de varias ciudades. Es capaz de detectar las siguientes situaciones: aglomeraciones de gente, personas que van en una dirección no permitida, elementos estáticos abandonados (equipajes, paquetes, basura…) y personas que están demasiado cerca de las vías. Otro ejemplo es PEACH [11]; diseñado para guiar de manera individualizada a los turistas en los museos. Su principal objetivo es mostrarle al turista sólo la información que le interesa, comunicándose con él de la manera más natural y sencilla posible.

- **Computación en la vestimenta:** El equipo de cómputo es un dispositivo pequeño que lleva el usuario consigo (normalmente integrado en la ropa) y que siempre está operativo y accesible. Ya que el dispositivo en la vestimenta siempre está operativo, se produce una sinergia entre el usuario y el dispositivo, caracterizada por un aprendizaje y una adaptación a largo plazo gracias a la interacción constante con el usuario. Estos sistemas son la manera en que el usuario puede acceder a los servicios ofrecidos por la AmI, independientemente de su localización. Algunos ejemplos de sistemas sobre la vestimenta desarrollados son los siguientes: asistente de traducción entre personas que hablan distintas lenguas, monitorización médica del usuario como el proyecto ***Wearable Eye Tracker*** [12] para la detección de esquizofrenia y otras enfermedades, etc.

- **Integración con la Web:** Como el ejemplo de COSM (anteriormente Pachube) [13], que es un servicio Web que permite compartir en tiempo real información proveniente de objetos equipados con *tags*, o sensores; ya sea una persona, o un objeto en cualquier parte del mundo (Figura 4). El objetivo es facilitar la interacción entre entornos remotos, tanto físicos como virtuales. COSM es parecido a Youtube, sólo que, en vez de compartir videos, este servicio permite monitorizar y compartir información del entorno en tiempo real proveniente de sensores con conexión a Internet. COSM hace de mediador entre entornos, sirviendo tanto de entrada (captura información de sensores remotos), como de

salida (sobre actuadores remotos). Un servicio así permitiría también crear un blog donde los datos de los sensores son los que "escriben" la propia página.

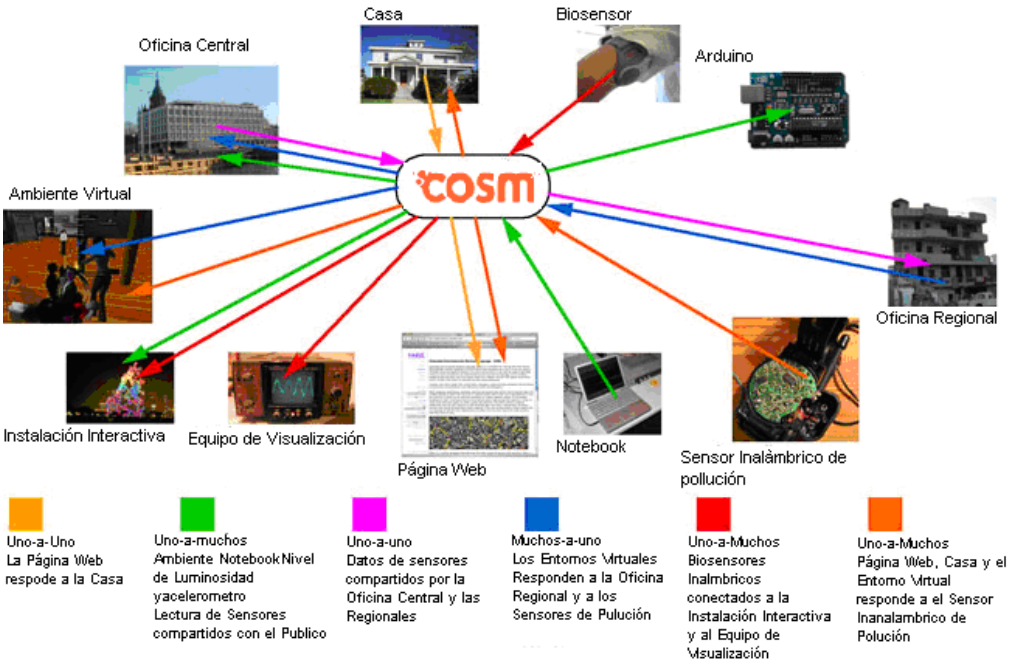


Fig. 4 – COSM: Integración de la Web con la AmI [13]

Las posibles situaciones donde se pueden aplicar entornos inteligentes son muy variadas, tal como se ha descripto hasta aquí. Sin embargo, todas estas aplicaciones de la AmI tienen una característica común: la necesidad de determinar la localización de los usuarios del sistema dentro del entorno inteligente. La localización de los usuarios es un aspecto importante, ya que, permite al sistema determinar los servicios que son más adecuados para el usuario según su ubicación. Además de la ubicación como se verá en capítulo 2 existen muchos otros aspectos del contexto que servirán para mejorar los beneficios que se pueden obtener de la AmI.

2. Aplicaciones Sensibles al Contexto

En capítulo anterior se so destacó como uno de los requisitos para el cumplimiento de la visión de la AmI al hecho de que las aplicaciones fueran sensibles al contexto, paro sin dar mayor de detalle de lo que esto significaba. En este capítulo se darán definiciones sobre que son las aplicaciones sensibles al contexto, pero para ello debemos definir previamente que significa contexto. Aunque la mayoría de las personas tiene una idea de lo que es el contexto a veces cuesta definirlo. Para llegar a una definición de lo que es el contexto en el campo de la informática se debe analizar cómo los investigadores lo han definido en el marco de las investigaciones que realizaron. Sin embargo esto no es una tarea fácil ya que existen varias definiciones.

2.1. Definiciones de Contexto

Las definiciones de contexto pueden ser clasificadas en dos grandes grupos: en un primer lugar están las enumerativas que intentan dar una definición por extensión, dando ejemplos de aquellas cosas que deben ser consideradas parte del contexto. En segundo lugar, están las definiciones sinonímicas, las cuales no son más que equivalentes de la palabra “contexto” como por ejemplo ambiente o situación.

Entre las definiciones enumerativas, podemos encontrar trabajo de Schilit y Theimer [14] que refiere al contexto como la ubicación, la identidad de las personas, los objetos cercanos, y los cambios de estos últimos. De la misma manera Brown [15] define el contexto como la ubicación, la identidad de la gente alrededor del usuario, la hora del día, la estación del año, la temperatura, etc. En su trabajo Ryan [16] define el contexto como la ubicación del usuario, el medio ambiente, la identidad y el tiempo. Por otro lado Dey [17] lo define como el estado emocional del usuario, su centro de atención, la ubicación y orientación, la fecha y la hora, los objetos y personas en el entorno del usuario. Estas definiciones que definen el contexto enumerando ejemplos de cosas que pueden ser parte del contexto, son difíciles de aplicar cuando se quiere

determinar si un tipo de información no aparece en la definición de contexto, no está claro cómo se puede utilizar estas definiciones para resolver ese dilema. Estas definiciones por ende no son suficientes cuando queremos incorporar nuevos aspectos del contexto a las aplicaciones.

Entre las definiciones sinonímicas algunos consideran que el contexto es el entorno del usuario, mientras que otros consideran que es el medio ambiente de la aplicación. Por ejemplo Brown [18] define contexto como los elementos del entorno del usuario que la computadora del usuario conoce. Franklin & Flaschbart [19] como la situación de los usuarios. Ward [20] ve al contexto como el estado de entorno de la aplicación.

Al igual que las definiciones enumerativas, las definiciones que emplean simplemente sinónimos de contexto son muy difíciles de aplicar en la práctica.

Las definiciones anteriores fueron evolucionando con el tiempo, tal es así que en [21] se hace notar que el contexto no es algo estático y que está en constante cambio durante la ejecución de la aplicación y que incorpora tres grandes aspectos:

- El entorno a nivel de computación, que incluye a los procesadores disponibles, conectividad, la capacidad de la red, costo computacional, etc.
- El entorno del usuario, que considera su ubicación, las personas cercanas, la situación social, etc.
- El entono físico, como por ejemplo la temperatura, el nivel de ruido, iluminación, etc.

Por otro lado Dey en [22] también evolucionó su definición de contexto, especificándolo como cualquier tipo de información que indique el estado físico, social o emocional del usuario. Así mismo, Pascoe [23] define contexto como el subconjunto de los estados físico y conceptual de interés para una entidad en particular.

Estas definiciones son demasiado específicas. El contexto lo es todo acerca de la situación de la aplicación y sus usuarios. Es imposible enumerar

todos los aspectos de todas las situaciones que son importantes, ya que esto va cambiando de situación a situación. En algunos casos, el medio ambiente físico puede ser importante, mientras que en otros puede ser completamente irrelevante.

En un intento por salvar los problemas de las definiciones anteriores Dey en [24] da la siguiente definición la cual es hoy en día la más aceptada por la comunidad científica: "*El contexto es cualquier información que puede ser usada para caracterizar la situación de una entidad. Una entidad puede ser una persona, un lugar o un objeto que es considerado relevante para la interacción entre el usuario y una aplicación, incluyendo al usuario y a la aplicación misma*".

La definición anterior muestra claramente que algunos tipos de contexto son más importantes que otros, como por ejemplo la ubicación, la identidad del usuario, la hora y la actividad que se está llevando a cabo. Esto se debe a que este tipo de información de contexto puede ser utilizada no solo para responder interrogantes directos (conocer donde está ubicado el usuario) sino que además puede ser usada en forma indirecta (saber cuáles son otros usuarios cercanos).

2.2. Definiciones de aplicaciones sensibles al contexto.

Las primeras definiciones de aplicaciones sensibles al contexto se referían y restringían a ser simplemente aplicaciones que eran informadas sobre el contexto para que se adaptaran a este [16]. Otras definiciones la muestran como la habilidad que poseen los dispositivos de detectar, sensar, interpretar y responder a los aspectos locales al ambiente de un usuario [27]. Dey [17] define la noción de *context-awareness* como el "*uso del contexto para automatizar un sistema de software, modificar su interface y proveer la máxima flexibilidad en términos de servicios*".

Otros autores señalan que una aplicación sensible al contexto es aquella que pueda variar o adaptar dinámicamente su comportamiento en base al mismo [25]. Salber [26] la define como "*aquella que tiene la capacidad de proporcionar la máxima flexibilidad de un servicio de cómputo basado en el*

censado del contexto en tiempo real". Otras definiciones pueden encontrarse en diferentes autores sobre el tema [21] y [27].

La definición de "sistema sensible al contexto" que adoptaremos para éste trabajo es la siguiente:"*sistema (aplicación) que hace uso de información del contexto para proveer información o servicios relevantes al usuario, donde la relevancia depende de actividad actual del sujeto"*. [22]. Entonces, se puede presentar al sistema como un proveedor de servicios e información, situándolo en un contexto dinámico, es decir la tarea que está realizando el usuario en un momento determinado.

2.3. Clasificación de Contexto

Consideraremos las siguientes categorías [26]:

- **Contexto geográfico:** información sobre la localización y la proximidad. Los datos de este tipo de contexto pueden ser tanto discretos (simbólicos) como escalares. Además, se puede distinguir entre un posicionamiento físico y lógico. Los datos de posicionamiento lógico son puramente simbólicos, por lo que requieren un conocimiento implícito para poder asignar un significado a los mismos. Por el contrario, el posicionamiento físico es el tipo de contexto de localización proporcionado por sistemas de posicionamiento como GPS, a través de valores escalares. Además, para poder manejar información de proximidad, es necesario distinguir a su vez entre posicionamiento relativo y absoluto.

- **Contexto físico:** Este contexto denota propiedades físicas del entorno como temperatura, ruido, luz, humedad, etc. Este contexto lo proporcionan sensores desplegados en un entorno determinado. El contexto físico puede describir propiedades de un objeto, como puede ser la presión arterial de un usuario. Los distintos artefactos físicos (edificios, objetos) y virtuales (redes, servicios) que rodean al sistema también pueden considerarse como parte del entorno físico. Es bastante usual combinar la información de contexto física con la geográfica, para conseguir elementos de contexto de nivel superior.

- **Contexto temporal:** Engloba cualquier tipo de información temporal: hora del día, fecha, semana, estación, etc. Normalmente, se requiere combinar este tipo de contexto con otros tipos (por ejemplo, obteniendo contextos espacio-temporales). Así, en muchos sistemas *context-aware* se emplea como una propiedad adicional de cada tipo de contexto específico, denotando la temporalidad de la información de contexto.

- **Contexto social:** Especifica las personas y las instituciones con las que un usuario interacciona. De este modo, describe grupos de gente como familias, círculos de amigos, compañeros de trabajo, etc. Frecuentemente, el rol que un usuario juega dentro del grupo se añade al contexto social. Un rol puede ser descrito a través de un nombre, un estado en este rol, las tareas que se esperan en este rol así como los distintos sub-roles que el rol puede comprender.

- **Contexto organizacional:** Se refiere a la necesidad de separar y estructurar procesos de negocio. Un contexto organizacional puede ser, por ejemplo, un departamento o el nombre de un proyecto.

- **Actividad y estado:** Es aplicable tanto a un usuario humano como a un objeto. Denota atributos simples como activo o pasivo (con respecto a una actividad) o información más detallada con respecto al estado de una cierta tarea o proceso.

- **Contexto de usuario:** Describe el contexto personal de un usuario. Es la combinación de los tipos mencionados anteriormente y puede contener otro tipo de datos como el perfil de usuario, los dispositivos que lleva un usuario en cada momento con sus características y configuraciones.

Para otros elementos de contexto, es muy difícil asignarles de forma única en uno de los tipos anteriores. Existen otras taxonomías más globales donde se agrupan los distintos elementos de contexto en cuatro dominios [27],

- **Dominio de usuario:** Comprende las características de los usuarios y su información personal como preferencias, actividades, etc.

- **Dominio físico:** Describe el entorno medible y sus propiedades físicas, incluyendo información espacio-temporal.

- **Dominio del sistema:** Incluye características de los dispositivos (capacidades de interfaces gráficas, memoria, velocidad de procesado, interfaces de red).

- **Dominio del objeto:** Contiene información sobre entidades sensibles al contexto como puede ser el mobiliario, puertas, ventanas, etc. y seres vivos como plantas y animales.

2.4. Clasificación de las aplicaciones sensibles al contexto

Conjuntamente con las definiciones de contexto y context-awareness, los autores desarrollaron una clasificación de este tipo de aplicaciones. Algunos trabajos presentan una caracterización basada dimensiones ortogonales, donde una dimensión indica si la aplicación debe recolectar información o ejecutar algún tipo de acción y la restante indica si la activación es manual o automática [21]. En base a estas dos dimensiones se definen cuatro tipos de aplicaciones:

- **Selección por cercanía:** Describe aquellas aplicaciones que brindan información de contexto al usuario en forma manual. Estas aplicaciones se basan en un paradigma de interacción en el cual la relevancia en que se presentan los objetos al usuario depende de la distancia a la que se encuentren.

- **Reconfiguración de contexto automática:** Son aquellas aplicaciones que brindan información de contexto al usuario en forma automática. Un ejemplo de este tipo de aplicaciones son las que indican que recursos se encuentran disponibles para el usuario, y se actualizan automáticamente ante los cambios de contexto.

- **Comandos contextuales:** Engloba a las aplicaciones que permiten ejecutar comandos basados en el contexto en forma manual. La disponibilidad de los

servicios depende del contexto del usuario, pero es éste y no el sistema quien decide ejecutar el comando.

- **Acciones activadas por contexto:** Define a las aplicaciones que ejecutan comandos automáticamente en base al contexto. Son modeladas como un conjunto de reglas de la forma if-then, indicando qué comando ejecutar en el caso que el contexto actual coincida con el condicional de la regla.

Otra definición basada en un conjunto de características genéricas que se indican a continuación [23]:

- **Sensado del contexto:** es el nivel más básico de context-awareness que puede tener una aplicación. Su única función es detectar los estados del ambiente y presentarlos al usuario en forma apropiada, aumentando efectivamente el sistema sensorial del usuario.

- **Adaptación contextual:** es la capacidad que muestra una aplicación de modificar su comportamiento en base al contexto, adaptándose así a la situación del usuario.

- **Descubrimiento de recursos:** denota la capacidad de descubrir y utilizar otros recursos que se encuentran disponibles en el contexto de la aplicación. Notar que esto es una actividad constante y dinámica, ya que depende fuertemente del contexto actual.

- **Aumento contextual:** es la capacidad de enriquecer al ambiente con información adicional. Dependiendo de la visión del usuario, podemos tomarlo como un aumento de la información física con información digital o viceversa.

Como corolario, se puede nombrar autores que proponen una categorización como una combinación de las ideas anteriormente mencionadas [24]:

- **Presentación,** ya sea de información o servicios. Sería equivalente a combinar la selección por cercanía y los comandos contextuales que aparecen en la taxonomía de Schilit.

- **Ejecución automática de servicios,** que sería equivalente a las acciones activadas por contexto de Schilit o a la adaptación contextual de Pascoe.

- **Etiquetado del contexto,** con el objetivo de ser examinado posteriormente. Es el equivalente a la noción de aumento contextual de Pascoe.

La particularidad de ésta última radica en que no distingue, entre información y servicios, considerando al descubrimiento de recursos ya no como una característica separada, sino como parte del proceso de selección de servicios en base al contexto.

Estas clasificaciones ayudan a determinar si una aplicación es efectivamente sensible al contexto o no. En caso de lo que sea, se puede especificar las principales características para a tener en cuenta cuando se avance en la etapa de desarrollo.

2.5. Modelos de representación de contexto

Los modelos de representación de contexto bien diseñados son indispensables para poder construir sistemas de este tipo. Los primeros modelos de contexto estaban diseñados para un tipo específico de aplicación o grupo de aplicaciones. Sin embargo, los modelos genéricos son en la actualidad los que más se desarrollan, y se intenta cada vez más un modelo lo suficientemente general y extensible para que no solo aplicaciones específicas puedan hacer uso de la incorporación del contexto a su funcionamiento. Más allá del modelo, son también necesarios lenguajes de consulta y representación de la información contextual así como los mecanismos de razonamiento que permitan el intercambio de información de contexto y la interoperabilidad de las aplicaciones. En este sentido para los sistemas sensibles al contexto actuales los requisitos para un modelo de contexto son [30]:

- **Composición distribuida:** En la actualidad, los sistemas pervasivos derivan de sistemas distribuidos, en los que no existe un punto central de

coordinación para la creación, despliegue y mantenimiento de los datos y los servicios.

- **Validación parcial:** Es deseable que los sistemas sean capaces de validar, al menos parcialmente, la información contextual contra el modelo de contexto en uso. Este requisito es particularmente importante por la complejidad de las interrelaciones contextuales, que hacen que cualquier modelo sea propenso a errores.

- **Información rica y de calidad:** La calidad de la información que los sensores y las fuentes de contexto proporcionan a un sistema, puede variar de unos a otros. Por ello, para llegar a un modelo genérico es necesario soportar la medida de la calidad y riqueza de la información proporcionada.

- **Información ambigua e incompleta:** El conjunto de la información contextual del sistema en entornos pervasivos es normalmente incompleta y/o ambigua. Los modelos que se empleen deben ser capaces de cubrir esta circunstancia.

- **Nivel de formalismo:** Describir la información contextual y las interrelaciones de forma precisa es un reto. Por ejemplo, para que un sistema sea capaz de realizar la tarea *"dame la temperatura del termómetro que está cerca de mí"* se necesita disponer de una definición precisa de los términos *"cerca"* y *"mí"*. Por tanto, es deseable que cada participante del sistema (se debe recordar que se habla de sistemas altamente distribuidos) sea capaz de interpretar la información intercambiada y su significado. Es decir, es necesario alcanzar un entendimiento compartido entre los distintos actores del sistema.

- **Aplicación a los entornos existentes:** Desde el punto de vista de la implementación es necesario que un modelo de contexto sea aplicable con la infraestructura y arquitecturas existentes, por ejemplo, en arquitecturas SOA (orientadas a servicio) basadas en servicios Web.

2.6. Clasificación de Modelos de Contexto.

Los modelos de contexto se pueden clasificar en función de las estructuras de datos que se utilizan para el intercambio de información contextual:

a. Modelos Clave-Valor *(key-value)*

Son los modelos más simples y se lo utiliza frecuentemente en sistemas basados en servicios distribuidos. Como ejemplo de este tipo de modelado es Hydra [31]. En estos sistemas, los servicios se describen a través de una lista de atributos simples de forma clave-valor. Por ejemplo para representar la información de temperatura se modelaría de la siguiente manera: *Key: Temperature- Value: 25* . Si bien los pares clave-valor son fáciles de manejar por los sistemas y se pueden incorporar a las arquitecturas de software actuales de forma sencilla, como contraparte no soportan una estructuración sofisticada de los datos, por lo que no pueden emplearse con algoritmos de recuperación de contexto eficientes. Por otra parte, carecen de un esquema de datos con la cual convalidar la información recuperada.

b. Modelos basados en esquemas de marcado

Todos los modelos basados en el marcado utilizan estructuras de datos jerárquicas formadas por etiquetas de marcado con atributos y su contenido. El contenido de las etiquetas se define normalmente de forma recursiva. Los más representativos de este tipo, son los perfiles, que se basan en una serialización de SGML (Standard Generic Markup Language). Algunos de éstos se definen como extensiones de los estándares CC/PP (Composite Capabilities / Preferences Profile) y perfiles de agentes de usuario UAProf (User Agent Profiles), que poseen la expresividad suficiente gracias a RDF/S y emplean serialización de XML. Estos modelos se limitan a extender y completar el vocabulario y procedimientos de CC/PP y UAProf para tratar con la complejidad y el alto dinamismo de la información contextual en comparación con los perfiles estáticos. CSCP (Comprehensive Structured Context Profiles) [32], emplea la flexibilidad de RDF/S para expresar estructuras naturales de los perfiles de información. Como puede observarse en la Figura 5, el esquema de

datos permite la validación de la información de contexto. Como desventaja encontramos necesidad de disponer información fiable a nivel de aplicación.

```
<?xml version="1.0" encoding="UTF-8"?><rdf:RDF>

xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:cscp="context-aware.org/CSCP/CSCPProfileSyntax#"
      xmlns="context-aware.org/CSCP/TemperatureProfileSyntax#" <TemperatureProfile
      rdf:ID="Temperature"> <temperature>

<value>25</value> <units>ºC</units> </temperature> </TemperatureProfile>
      </rdf:RDF>
```

Fig. 5 – Código CSCP para valor de temperatura

c. Modelos gráficos

Estos modelos emplean UML (Unified Modeling Languaje). A través de diagramas UML, como herramienta de modelado del contexto, ya que posee una estructura suficientemente genérica [33]. La ventaja de los modelos gráficos es a nivel estructural, dado que puede describir la estructura del conocimiento contextual de forma sencilla, permitiendo una aplicación rápida y la validación parcial de los datos [34]. Si bien hoy en día existen herramientas de software que pueden traducir estos modelos a código, este tipo de modelado es aprovechado mayormente a nivel conceptual por los desarrolladores.

d. Modelos orientados a objetos

Estos modelos intentan incorporar los beneficios de las arquitecturas orientadas a objeto como respuesta los problemas derivados de la dinámica de la información del contexto en entornos ubicuos. En estos modelos los detalles del procesamiento de la información de contexto se encapsulan a un nivel de objetos y por tanto a otros componentes, y el acceso a la información contextual se realiza a través de interfaces.

Un ejemplo representativo se presenta a través de las indicaciones (del término inglés cues) [35]. El concepto de *cues* proporciona una abstracción de

los sensores físicos y lógicos. Una *cue* representa una función que toma como entrada el valor de un único sensor en un tiempo determinado y proporciona una salida simbólica, en la que se definen una serie de posibles valores. La salida de cada *cue* depende de un sensor único, pero diferentes *cues* pueden basarse en el mismo sensor. El contexto se modela como una abstracción de todas las *cues* disponibles. Por tanto, las *cues* son objetos que proporcionan información contextual a través de sus interfaces, escondiendo los detalles de la determinación de los valores de salida.

Una de las ventajas de estos modelos es que se pueden aplicar fácilmente en entornos distribuidos. Otra ventaja es la validación y la medida de la calidad de la información proporcionada también son posibles. Por último, se alcanza un elevado nivel de formalismo, ya que se emplean interfaces bien definidas para el acceso a la información contextual.

La posible desventaja es posee unos requisitos computacionales elevados, por lo que su aplicabilidad en sistemas pervasivos resulta a complicado a veces.

e. Modelos basados en la lógica

En estos modelos, una lógica define las condiciones en las que una expresión o un hecho pueden ser derivados (a este proceso se le conoce como razonamiento o deducción) a partir de un conjunto de otras expresiones o hechos. Para describir las condiciones como un conjunto de reglas, se aplica un sistema formal. Por lo tanto, el contexto en estos modelos queda definido a través de hechos, reglas y expresiones. Normalmente, la información contextual se añade, se actualiza y se borra en términos de hechos o deducciones de las reglas. Por tanto, todos los modelos basados en la lógica poseen un elevado nivel de formalismo.

Un ejemplo de este tipo de modelos es el Sensed Context Model [36], donde se emplea lógica de predicados de primer orden como una representación formal de proposiciones contextuales y relaciones.

Estos modelos permiten ser distribuidos, pero la validación parcial es difícil de mantener. Aunque poseen un nivel de formalismo elevado, sin validación la especificación de una base de conocimiento contextual es muy propensa a errores. Además, su aplicabilidad a sistemas pervasivos es complicada, ya que los requisitos computacionales del razonamiento lógico son demasiado elevados.

f. Modelos basados en ontologías

Las ontologías son instrumento perfecto para especificar conceptos e interrelaciones, ya que proporcionan los formalismos para proyectar entidades del mundo real en construcciones de datos entendibles por las máquinas. De esta forma, las ontologías proporcionan un mecanismo uniforme de especificar los conceptos, subconceptos, relaciones, propiedades y hechos del modelo y a la vez proporcionan los medios para la compartición del conocimiento del contexto y el rehúso de información.

Uno de los enfoques más prometedores de modelos basados en ontologías se propone en el sistema CoBrA [37]. En este sistema se describe la ontología CoBrA-ONT [38] que permite caracterizar entidades como personas, lugares o distintos tipos de objetos junto con sus contextos para el desarrollo de salas de reuniones inteligentes. El sistema CoBrA emplea una arquitectura de agentes basada en fuentes de contexto y gestores de contexto. La ontología de CoBrA permite que el gestor de contexto comparta el conocimiento contextual con otros agentes (fuentes de contexto) y razonar sobre la información de contexto. Para la definición de la ontología, se emplea el lenguaje OWL ya que se trata de un lenguaje mucho más expresivo que RDF o RDFS. En la Figura 6 se muestra una lista completa de las clases y propiedades definidas en la ontología COBRA-ONT.

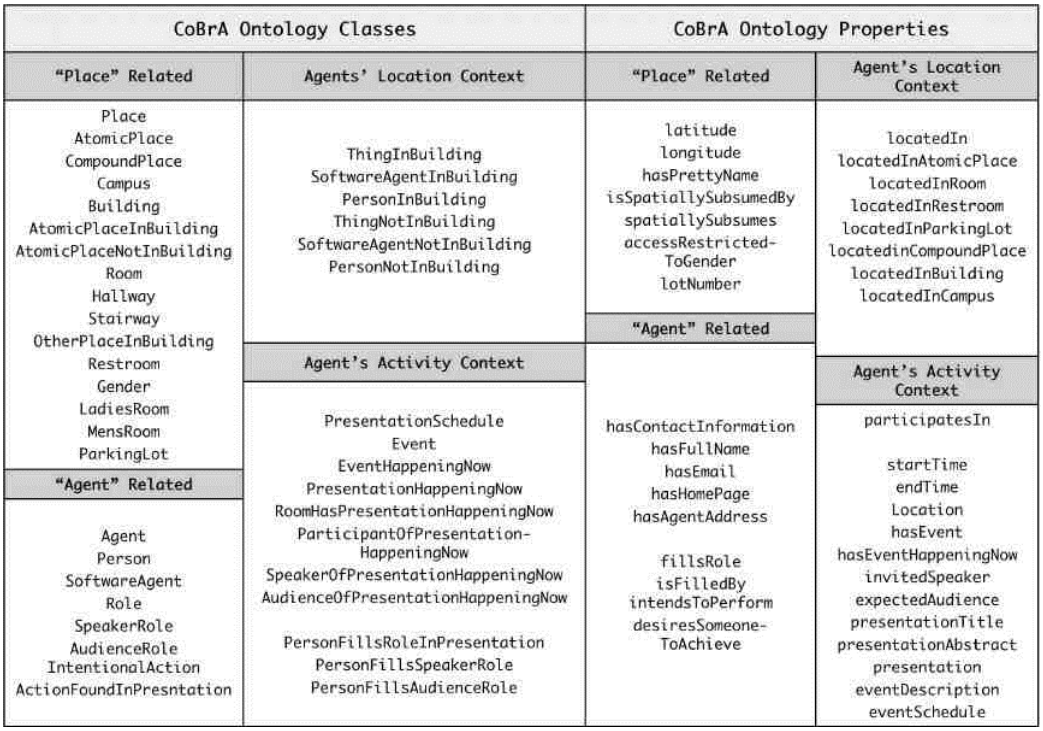


Fig. 6- Clases y Propiedades definidas en COBRA-ONT [37]

3. Mashups

Nadie podía prever hace 20 años el importante papel que Internet está jugando hoy, convirtiéndose en una de las piedras angulares de nuestra sociedad, siendo elemento fundamental en cuanto al soporte de las interacciones sociales y económicas en una escala global. En el futuro, el rol jugado por Internet será más evidente a medida que más actividades basadas en la red lleguen para quedarse en nuestra vida diaria.

El término "Web 2.0" [39] se ha convertido en soporte natural y fundamental con referencia a los patrones de diseño y modelos de negocio relacionados a tecnología de Internet y su utilización. El concepto hace referencia a una Internet basada en servicios, en la cual los clientes colaboran y comparten información en línea, aportando así una nueva forma de interacción social. Wikipedia es uno de los mejores ejemplos de sitios web que han nacido, quién sabe si al amparo de este concepto, o como motor de él. El contenido de Wikipedia es producido, editado, organizado y traducido por los propios usuarios [40] La Web 2.0 además incluye una gama de servicios mejorados, como ser los, wikis, blogs, bittorrents, servicios web y la sindicación de contenidos

Una de las técnicas más destacadas en el ámbito de Web 2.0 la compone el método de "mezcla" o *mashup*. Este término se origina en la práctica de mezclar muestras de dos o más pistas o fuentes de sonidos para producir otras nuevas. Esta técnica se ha generalizado y llevado a otros campos, dando la idea básica de desarrollar un nuevo trabajo de la mezcla de dos o más, ej: el dominio digital, el dominio de vídeo y el dominio Web.

A pesar de que su significado es bastante claro, no existe una definición oficial de *mashups*, por se utilizará una definición basada en la presentada en [41] que dice *"Un Mashup es una aplicación basada en Web que se crea mediante la combinación y procesamiento de recursos en línea de terceros, que contribuyen con datos, presentación y funcionalidad."* Como resultado, un

mashup proporciona un nuevo recurso, mediante el uso de aplicaciones de terceros que hacen de proveedores de contenido, de las interfaces de programación (API) y las tecnologías abiertas, como por ejemplo, Ajax y PHP, y la sindicación como la que se consigue con los *feeds* RSS o ATOM. Además, puesto que el contenido puede ser obtenido de varias fuentes, han surgido empresas intermedias que actúan como intermediarios de contenido [42].

3.1. Arquitectura de un *Mashup*

Un modelo arquitectónico de *mashup* consta de tres participantes diferentes: la API/proveedores de contenido, el sitio de alojamiento del *mashup* y el navegador Web del consumidor [43] y [44] La arquitectura se muestra en la Figura 7 y a continuación se describen sus componentes.

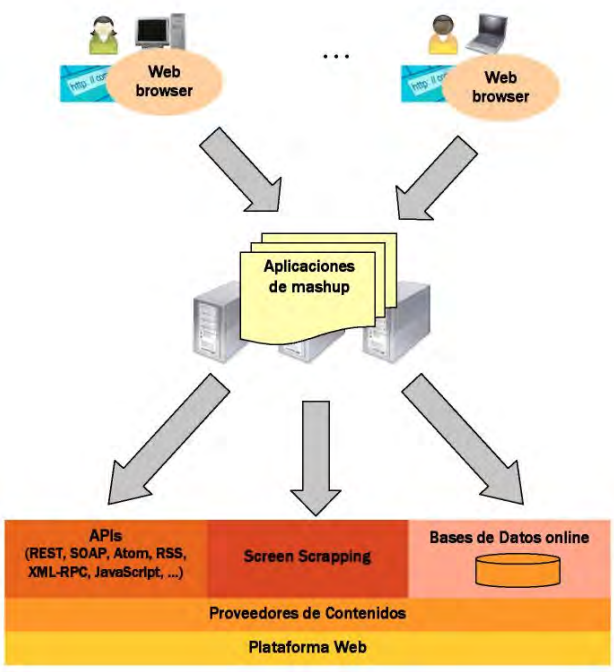


Fig. 7 – Arquitectura de un *Mashup* [43]

a) La API / proveedores de contenido. Estos son los proveedores de contenido (a veces heterogéneos) de los cuales se desea hacer el *mashup*.

Para facilitar la recuperación de datos, los proveedores suelen dar acceso a sus contenidos a través de protocolos de Internet tales como REST, los Web Services o feeds RSS / Atom. Sin embargo, muchos datos posiblemente interesantes existen en fuentes que todavía no están disponibles en uno de los protocolos anteriores o que los proveedores no creen convenientemente publicarlos en forma de API. Es por ello que algunos *mashups* extraen contenidos de sitios como Wikipedia, TV Guide, y prácticamente todos los sitios de gobierno y las Web de dominio público, utilizando una técnica conocida como *screen scraping* [45].

b) El hosting del *mashup* (aplicaciones *mashup*). Es donde el *mashup* se encuentra alojado, pero no porque la lógica del *mashup* resida un cierto espacio significa que sea ejecutado en el mismo lugar. Por una parte, los *mashups* pueden utilizarse de manera similar a las aplicaciones Web tradicionales utilizando un servidor de contenido dinámico utilizando alguna de las tecnologías de generación de contenido del lado del servidor, como Java Servlets, CGI, PHP o ASP. Alternativamente, el contenido del *mashup* se puede generar directamente en el navegador del cliente a través de secuencias de comandos en el cliente utilizando por ejemplo JavaScript o *applets*. La lógica del lado del cliente es a menudo la combinación de código incrustado directamente en las páginas web del *mashup*, así como también secuencias de comandos de las APIs o *applets* (proporcionados por los proveedores de contenidos) a las cuales hacen referencia a éstas páginas. Los *mashups* más recientes utilizan una tecnología llamada Aplicaciones de Internet Enriquecidas (RIA) por sus iniciales en inglés, lo que significa que están muy orientados hacia la experiencia interactiva del usuario. Los beneficios del lado del cliente son menos sobrecarga del servidor *mashup* (los datos pueden ser consultados directamente desde el proveedor de contenidos) y una experiencia de usuario más cómoda (las páginas pueden solicitar actualizaciones de parte de sus contenidos sin tener que cargar nuevamente la página completa). Por ejemplo, la API de Google Maps tiene disponible el acceso a través del navegador utilizando JavaScript, y

es un ejemplo de *mashup* con lógica del lado del cliente. A menudo un *mashup* utiliza una combinación de lógica del lado del servidor y cliente para conseguir una agregación de datos más eficiente.

c) **El navegador Web de los consumidores.** Es donde la solicitud se representa gráficamente y la interacción del usuario con el *mashup* se lleva a cabo. Como se describió anteriormente, los *mashups* utilizan a menudo la lógica del lado del cliente para montar y componer el contenido del *mashup*.

3.2. Clasificaciones de *Mashups*

Existen varios modelos de categorización de *mashup* [46, 47, 48, 49, 50]. En resumen, estos pueden clasificarse en base a las siguientes cuatro preguntas: **¿Qué?, ¿Dónde? , ¿Cómo? y ¿Para quién?**

1. ¿Qué?

Dependiendo del tipo de recurso que combinan o integran los *mashups* pueden ser clasificados en una de las siguientes tres categorías: presentación, datos o funcionalidad de la aplicación.

a) ***Mashup de presentación*** se centra en la recuperación de la información y el diseño de las diferentes fuentes de Internet, sin relación con los datos subyacentes y funcionalidad de la aplicación. Para este tipo de pre-reproductores *mashup* construido, simplemente arrastra y se suelta componentes o *widgets* en una interfaz de usuario común. Por lo general, la creación de un *mashup* de este tipo requiere poco o ningún conocimiento de lenguajes de programación.

b) ***Mashup de datos*** combina los datos proporcionados por diferentes fuentes (por ejemplo, servicios Web, HTML plano, etc.) en una sola página de contenido (por ejemplo, para una

ciudad dada, se puede obtener la combinación de diferentes servicios como el pronóstico del tiempo, eventos y fotos).

c) ***Mashup de funcionalidad*** combina los datos y la funcionalidad de aplicaciones de diferentes fuentes, en nuevo servicio. Las funcionalidades son accesibles a través de una API.

Otra clasificación que se puede ver en [47] categoriza a los *mashups* como: *mashups* de mapas (Ej: Google Map), *mashups* de Foto-/Video- (Ej: Flickr-Yahoo), *mashups* de Compras y *mashups* de noticias entre otros. Esta clasificación es un refinamiento de las comentadas anteriormente. La Figura 8 indica la utilización de los 10 tipos de *mashups* más destacados actualmente.

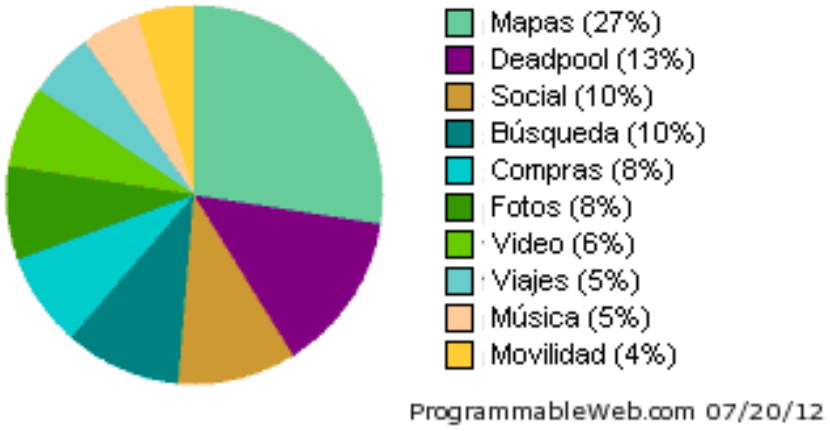


Fig. 8 – Clasificación de *Mashups* según su funcionalidad [47].

2. ¿Dónde?

Los *mashups* pueden también clasificarse dependiendo de dónde se realiza la integración de las fuentes o APIs:

a) **Del lado del Cliente** más enfocado en la mezcla o composición de información con imágenes del lado del navegador, utilizando principalmente JavaScript para su implementación. Un ejemplo

clásico de este tipo de *mashups* es aquél en que se combina el servicio de Google Maps, con otro servicio, por ejemplo de precios de casas, que muestra en una sola pantalla datos de inmueble como el precio, número de habitaciones, comodidades, etc. y la ubicación donde se encuentra una casa en venta.

b) **Del Lado del servidor** la integración y manipulación de la información suceden en el servidor. Su uso principal es interactuar con información de diferentes sistemas para generar vistas necesarias para la toma de decisiones. Este tipo de *mashups* es capaz de interactuar con bases de datos para generar valor añadido.

3. ¿Cómo?

Los *mashups* pueden clasificarse dependiendo de la modalidad de los recursos están integrados o combinados.

a) ***Mashup de extracción*** puede ser considerado como un contenedor de datos, el análisis de los recursos de diferentes fuentes y la fusión de estos recursos a una sola página de contenido.

b) ***Mashup de flujo*** en donde que el usuario personaliza el flujo de recursos de la página Web y la combinación de recursos de diferentes fuentes. Los recursos son transformados e integrados dentro de la aplicación *mashup*.

4. ¿Para quién?

Las diferentes herramientas de *mashup* pueden ser utilizadas no solo para construir aplicaciones web híbridas que combinan contenidos de diferentes fuentes, sino también para distinguir el grupo destinatario que se trate. En este contexto, *mashups* se pueden clasificar en *mashups* para consumidores y *mashups* empresariales, también conocidos como *mashups* de negocios.

a) ***Mashup para consumidores:*** Se destina para uso público y combina los recursos (por ejemplo, el diseño o los datos) de diferentes fuentes públicas o privadas en el navegador y los organiza de una manera simple basada en la interface de usuario.

b) ***Mashup empresarial:*** **C**ombina varios recursos (por ejemplo, datos y funcionalidad de la aplicación) de los sistemas en un entorno empresarial. Estos *mashups* combinan datos y funcionalidades de distintas aplicaciones, por ejemplo, los diferentes sistemas, ERP, CRM o SCM, a fin de responder a sus objetivos. La creación de *mashups* empresariales requiere considerar la seguridad y las políticas de empresa. Estos *mashups* ofrecen una manera rápida para que se fusionen recursos internos y externos de diferentes fuentes sin un intermediario.

3.3. Técnicas y Tecnologías de Soporte

Del el conjunto de las tecnologías implicadas en las plataformas Web, se destacan algunas que habitualmente son usadas en la construcción de aplicaciones *mashup*. Además de estas tecnologías, existen técnicas ya conocidas que también pueden ser usadas. A continuación se enumeran y describen tecnologías y técnicas utilizadas en la construcción de mahups.

1. **APIs:** Las APIs usadas en el contexto de las aplicaciones *mashup* recurren a *web services*. Un *web service* es definido por el W3C Consortium [51] como un sistema de software diseñado para soportar interoperabilidad entre máquinas, sobre una red de computadoras. Los *web services* facilitan servicios en una red usando tecnologías como XML y HTTP. La interacción con un *web service* se hace a través de mensajes SOAP. La interoperabilidad entre máquinas se realiza a través de WSDL (Web Services Description Language) [52]. Una definición WDSL de un servicio web incluye una descripción de funcionalidades, codificación de datos, protocolos de comunicación y localización del servicio. Las APIs deben funcionar de acuerdo con principios de arquitectura conocidos como REST (Representational State Transfer).

REST representa una arquitectura de software aplicada a sistemas hipermedia distribuidos como la Web. Esta arquitectura fue introducida en la tesis doctoral de Roy Fieldind [53] uno de los principales autores del protocolo HTTP en el año 2000. El término REST también es usado para describir interfaces de transmisión de datos usando HTTP y mensajes SOAP. El término generalmente es asociado a interfaces que recurran a las tecnologías XML+HTTP.

2. **SOAP** [54], originalmente un acrónimo de Simple Object Access Protocol, es un protocolo que especifica la estructura de la información que hace parte de la implementación de un *web service*. Con la evolución de este protocolo, los términos Simple y Object dejaron de tener sentido, pasando SOAP a ser simplemente el nombre del protocolo. El formato de los mensajes es soportado por XML, estando su transmisión asegurada por RPC (Remote Procedure Call) o HTTP.

3. Los **web feeds**[2] facilitan que un usuario de la plataforma web pueda utilizar los contenidos de un sitio Web o blog, sin tener que visitarlo. Cuando hay un nuevo contenido el usuario del feed es inmediatamente notificado. Existen tres especificaciones para la creación de web feeds: a) RSS **1.0 - RDF** Site Summary 1.0; b) RSS 2**.0 - Real**ly Simple Syndication 2.0; c) Atom.

4. **Ajax:** [58], acrónimo Asynchronous Javascript And XML, consiste en el uso conjunto de varias tecnologías como ser XHTML, CSS, DOM, XML y Javascript las cuales permiten hacer a los sitios web más interactivos y dando una experiencia de "escritorio" a los usuarios Ajax además permite una comunicación asincrónica con el servidor permitiendo que parte de una página web se actualice sin tener que volver a cargarla por completo. Está presente en algunas herramientas de creación de *mashups.*

[2] listas de actualización de contenido de un determinado sitio Web, escritos con especificaciones basadas en XML

5. **Screen Scrapping:** Las técnicas de *screen scrapping* representan los procesos a través de los cuales una herramienta extrae información de un proveedor de contenidos mediante la lectura de sus páginas web. Se trata de transformar información presentada de tal manera que es entendida por usuarios humanos, en datos que puedan ser utilizados en la construcción aplicaciones *mashup*. Este tipo de solución presenta ciertos inconvenientes como ser: que quien aplica la técnica utiliza un modelo de contenido sin que exista un contrato con el proveedor de la información. Si él proveedor altera el formato de presentación de la información de su sitio, llevaría al consecuente fallo de la herramienta de *scrapping*. Un ejemplo de este tipo de software es el screen-scrapper [59].

6. **Web semántica y RDF:** La incompatibilidad entre los contenidos para consumo humano y contenidos para consumo automático (por máquinas), puede ser solucionado por la Web Semántica. Este concepto, anterior al de la Web 2.0, permite la interrelación de datos, añadiéndoles significado, de tal forma que éstos puedan ser entendidos por usuarios humanos y por máquinas. La integración de tecnologías XML, RDF (Resource Description Framework) [60] y ontologías, entre otras, permiten la creación de servicios que garanticen la interoperabilidad y la cooperación, así como potenciar el desarrollo de aplicaciones de *mashup*.

3.4. Herramientas de creación de *Mashups*

Actualmente existen herramientas de desarrollo de aplicaciones de *mashup*. De forma general esas aplicaciones son bastante sencillas. Tienen como objetivo poder ser utilizadas por usuarios sin grandes conocimientos técnicos. Algunos ejemplos de estas herramientas son:

a) **Yahoo Pipes** [61] - Yahoo lanzó esta aplicación destinada a usuarios sin conocimientos de programación, popularizándose rápidamente. Funciona en un abordaje arrastrar y soltar que permite el acceso a servicios como Flickr, Google Base, Yahoo Local y búsquedas de Yahoo. Pueden

también contener *feeds* con herramientas que filtran, enlazan y restringen noticias, fotografías y otros contenidos. Una característica interesante, para los nuevos usuarios de la herramienta, es poder usar como base los miles de ejemplos creados por la comunidad de este servicio, además de permitir la clonación de soluciones ya existentes.

b) **DERI Pipes [62]:** Es una plataforma para producir *mashups* extensibles, embebibles y de código abierto, inspirado en Yahoo's Pipes. Deri Pipes es un motor y un ambiente gráfico para desarrollar trasformaciones de datos de la web y *mashups*. Soporta RDF, XML, Microformatos, JSON y flujos binarios. Se lo puede usar solo o embebido en una aplicación del usuario. Trabaja como una herramienta para crear *mashups* a través de líneas de comando y soporta SPARQL, XQUERY y algunos otros lenguajes de script. Se lo puede extender según sea necesario. Las salidas son producidas como flujos de datos en formato XML, RDF y JSON entre otros que ser usado por otras aplicaciones. Sin embargo, cuando se lo utiliza solo desde un browser, este provee una GUI para que usuario final ingrese los parámetros y vea los resultados.

c) **SnapLogic [63] –** Es una plataforma para la creación de *mashups* empresariales basada en una arquitectura REST potente y modular que permite la integración de datos, ya sea en la nube o en las instalaciones. Permite conectar aplicaciones en locales y en la Web a través de un motor estándar. Posee una GUI potente y fácil de usar. Su arquitectura 100% orientada a la web permite que se puedan utilizar para las consultas a las bases de datos en lenguajes tradicionales como con los para la web. Tiene soporte total para XML, JSON, Atom y otros.

Una lista de algunas herramientas, en el ámbito de los *mashups* empresariales puede ver en [64]. Otros ejemplos de herramientas y plataformas de desarrollo son WSO2 Mashup Server [65] u Open Mashups Studio [66].

4. Plataformas para el desarrollo de *Mashups* Sensibles al contexto

En este capítulo se analizan, en cuanto a funcionalidad, componentes e implementación, tres casos/proyectos existentes de plataformas para construir *mashups* sensibles al contexto. El análisis realizado fue presentado por el autor del presente trabajo en [67] y ampliado en [68]. Las plataformas que se analizaron fueron:

4.1. El Caso del Proyecto Deusto Sentient Graffiti (DSG)

El proyecto Deusto Sentient Graffiti [69, 70, 71] busca la convergencia de la computación móvil y la computación ubicua con la web 2.0 para configurar lo que hoy día se llama Web Ubicua (UW). Esta última se puede definir como una infraestructura de Internet omnipresente en la que todos los objetos físicos son recursos accesibles a través de una URI. Estas URIs pueden proporcionar información y servicios que enriquezcan la experiencia de los usuarios en su contexto físico.

El principal objetivo de DSG es disminuir la barrera que implica desarrollar e implementar sistemas sensibles al contexto y que se puedan proporcionar y consumir servicios inteligentes, ya sea en ambientes exteriores o interiores. Su funcionamiento se basa en las anotaciones espontáneas o "Graffitis" (en formato de documento XML) que realiza una comunidad de usuarios sobre objetos, lugares o incluso personas, con contenido Web multimedia y/o servicios que pueden ser descubiertos y utilizados a su vez por otros usuarios móviles cuyos atributos contextuales coincidan con las de las anotaciones.

Los autores plantean que su trabajo no tiene solamente el propósito de diseñar y desarrollar el sistema DSG como un ejemplo práctico de la web ubicua, que se acerque a la visión de la AmI, sino que además, la principal motivación es proveer un marco de trabajo reutilizable para construir aplicaciones sensibles al contexto en la vida real. Estas aplicaciones presentan los siguientes requerimientos funcionales comprendidos por DSG:

- Modelo de todos los objetos físicos o regiones espaciales cuya información o servicios puedan ser consumidos.

- Poner a disposición de los usuarios sólo las anotaciones asociadas a los recursos cercanos disponibles en sus condiciones contextuales actuales o requisitos de filtrado deseado.

- Facilitar la interacción explícita controlada por el usuario con el objeto inteligente y regiones espaciales encontradas por un usuario móvil, tanto en modo PUSH como PULL.

4.1.1. Funcionamiento

El proyecto está desarrollado siguiendo una arquitectura de cliente\servidor en la que los usuarios de dispositivos móviles o a través de un navegador web tradicional ejecutan un software llamado cliente DSG, mientras que el servidor se ejecuta el software servidor DSG, el cual almacena, indexa y empareja las anotaciones almacenadas con el contexto actual de los usuarios publicado por los clientes DSG.

Desde el punto de vista del usuario la funcionalidad consiste en dos procesos a saber:

- Anotación de Graffitis: Los usuarios de los clientes DSG anotan objetos o regiones espaciales las cuales consisten de:

    - Descripciones: Como ser contenido multimedia o la URL de un servicio web.

    - Palabras claves: Las cuales describen anotaciones sobre recursos y permiten su clasificación.

    - Atributos contextuales: Los cuales definen las condiciones que deben cumplir los consumidores de las anotaciones. Algunos de estos atributos los establecerá automáticamente el cliente DSG (Ej: quién creó la anotación, dónde y cuándo),

mientras que otros deben ser establecidos explícitamente por el usuario (rango ubicación donde se debe ver la etiqueta, quién la puede ver, cuándo y por cuánto tiempo).

- Descubrimiento y Consumo de Graffitis: Los usuarios, con el cliente DSG corriendo en sus dispositivos móviles o a través de un browser web, puede moverse virtual o físicamente a través de un ambiente "anotado" navegando y consumiendo las anotaciones disponibles que se emparejen con el contexto actual del usuario, su perfil y/o preferencias. Esta interacción se puede dar de forma explícita o implícita. La forma explícita (PULL) se da cuando los usuarios interactúan con la aplicación solicitando las anotaciones disponibles. En la forma implícita (PUSH) el sistema alerta al usuario cuando existen nuevas anotaciones disponibles que concuerdan con sus atributos contextuales actuales.

Desde el Punto de vista del Sistema: DSG pude verse tanto como una folksonomía sensible al contexto y como un *mashup* sensible al contexto. Como una folksonomía sensible al contexto, en DSG se establece una clasificación espontanea de los objetos y las regiones espaciales anotadas y las relaciones entre estos. Por lo tanto, es posible vincular las anotaciones de los recursos y compartir todas o un subconjunto de palabras claves. Como *mashup* sensible al contexto, los clientes de DSG móviles y basados en web pueden combinar la información geográfica en forma de mapas (por ejemplo, obtenidos a partir de Google Maps) y la información suministrada por el servidor DSG en las descripciones.

4.1.2. Componentes de la Plataforma

La plataforma DSG presenta los siguientes componentes los cuales se pueden observar en la Figura 9.

- Administrador de Graffiti & Web Server: administra todos los graffitis virtuales publicados, junto con su contenido y metadatos (palabras claves y atributos contextuales) que se encuentran almacenados en una base de datos relacional, denominada Repositorio de Graffitis.

- Disparador y Notificador de Graffitis: Es un motor de inferencia basado en reglas en el back-end que rellena su base de conocimientos con los cambios de atributos contextuales recibidos de los clientes DSG, y deduce los conjuntos de anotaciones activas que tienen que ser notificadas a los usuarios y dónde tienen que ser mostradas. La inferencia de anotaciones es soportada por un conjunto de reglas genéricas almacenadas en una base de reglas que es el núcleo de la inteligencia de DSG. La separación de la inteligencia del sistema en un conjunto de normas es un enfoque muy flexible para poner a punto continuamente el comportamiento del sistema. Estas normas no sólo determinan que nuevas anotaciones se necesitan entregar a un cliente DSG, además están a cargo de la recolección de basura, es decir de anotaciones que ya hayan caducado (aquellas que se consumen un número de veces previamente especificado o cuya tiempo de vida haya expirado). Este componente permite a un cliente DSG funcionar en un modo PUSH, sin intervención del usuario una vez por se ha establecido sesión entre el cliente y el disparador que se encuentra en el servidor DSG.

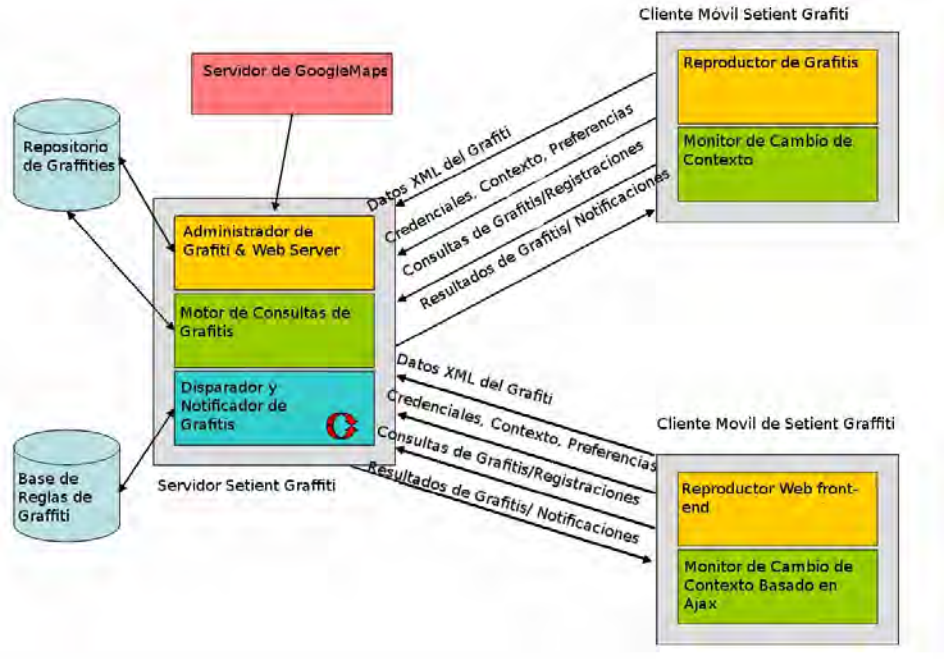


Fig. 9 – Componentes de DSG

- Motor de Consultas de Graffitis: Este componente del back-end es responsable del procesamiento de consultas y de la recuperación de la anotaciones sensibles al contexto realizadas por los clientes DSG al Repositorio de Graffitis. Este componente permite a un cliente DSG funcionar en una forma PULL, la comunicación con el back-end sólo se realiza bajo pedido explícito del usuario.

- Monitor de Cambio de Contexto: Se ejecuta en el dispositivo móvil o navegador web y su función es la de vigilar los cambios significativos en el contexto del usuario y que hacer las veces de "caché" de la información del cambio de contexto para su transmisión en consultas sobre anotaciones o para alimentar disparador de Graffitis cuando se opere en forma PUSH. Uno de los objetivos principales de este componente es reducir la cantidad de datos transferidos entre la aplicación móvil y el servidor DSG a fin de disminuir los costes de comunicación y aprovechar al máximo la interactividad del sistema.

- Reproductor de Graffitis o Web front-end: Muestra graffitis virtuales en el dispositivo del usuario, que puede ser una computadora (aplicación web en un navegador web, mostrando un mapa combinado con marcadores que indican al contexto recuperados graffitis) o un dispositivo móvil (clientes Java ME o Compact. NET, tanto con una vista de mapa o una lista de graffitis disponibles en un determinado lugar). A parte de reproducir los graffitis (proporcionar contenidos multimedia o un servicio web), también permite la creación, eliminación y edición de ellos.

4.1.3. Implementación

Para su implementación DSG utiliza una sintaxis XML llamada GraffitiXML que se usa para especificar contenido. Está compuesto por dos partes:

- Una Cabecera que consiste de atributos generales como id, titulo, y descripción, un conjunto palabras claves que identifican el propósito de graffiti y algunos atributos contextuales. Estos últimos pueden ser

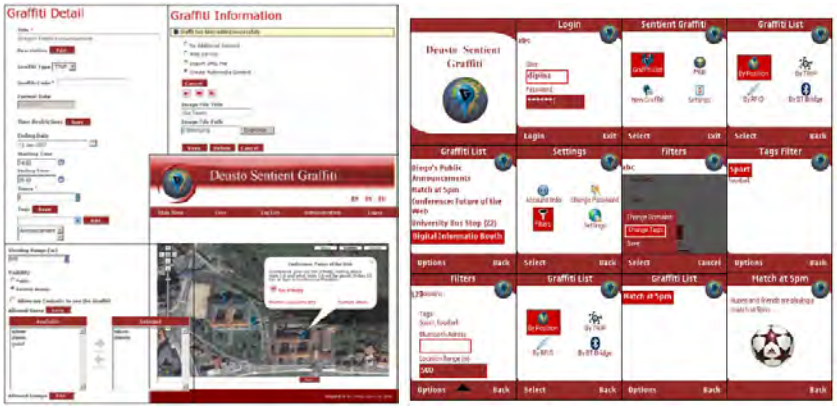
implícitos, es decir generados automáticamente por los clientes DSG o explícitos creados intencionalmente por los usuarios.

- El Cuerpo contiene archivos multimedia en formato SMIL [72] o la URL de un servicio web.

Como todo sistema basado en Web 2.0, como Google maps o Flickr, DSG provee una API basado en HTTP que permite a programadores de terceras partes desarrollar aplicaciones que se beneficien de su funcionalidad. La API requiere que los clientes puedan enviar solicitudes http POST.

La implementación del servidor DSG está basada en Java y se ejecuta sobre el servidor de aplicaciones Tomcat utilizando JAVA EE. Además utiliza los marcos de trabajo, Tapesty, Hibernate, y Java Expert Systems Shell y como base de datos MySQL.

En cuanto a la implementación del cliente DSG basado en web se utiliza el marco de trabajo Dojo AJAX. En la Figura 10 (a) se puede ver una captura de pantalla de la interfaz que ilustra cómo crear graffitis, asignar restricciones y verlas.



(a) (b)

Fig. 10 - Interface Web (a) y para teléfonos móviles Java ME (b) de DSG

Para la implementación del cliente DSG para dispositivos móviles se utiliza Java ME, la cual permite la creación, búsqueda, descubrimiento y reproducción de los graffitis en los dispositivos. En la Figura 10 (b) se muestran unas capturas de pantalla de un teléfono móvil en las que se puede ver la

funcionalidad. Primeramente con la ayuda del GPS se descubren los graffitis que están alrededor del usuario según la posición, seguidamente se presenta la vista de un mapa de los graffitis descubiertos y finalmente se repite la búsqueda según la posición después de haber establecido la posición de la anotación y los filtros de localización.

4.2. El Caso del Proyecto Personalized Location based Services over Mobile Architectures (PLASMA)

Los autores de PLASMA [73, 74] presentan un marco de trabajo experimental que permite a los usuarios construir, desplegar y distribuir servicios móviles personalizados. Este marco da la posibilidad de integrar información de diversos sistemas, contenido estático y dinámico de varias fuentes de la Web, y las preferencias de los usuarios. Permite combinar toda esta información en forma de *mashup* para dar soporte a la provisión de servicios móviles personalizados y cumplir la visión de la AmI.

4.2.1. Funcionamiento

Para componer un nuevo servicio, la aplicación que se ejecuta en el dispositivo móvil del usuario tiene que identificar primero todos los componentes de servicios disponibles en el servidor, mediante el uso de la API correspondiente. A estos componentes de servicio se les pasa como parámetros la información de contexto, que ha sido recogida por el dispositivo móvil, y el nivel de precisión/privacidad que el usuario ha seleccionado.

Hasta aquí, la configuración del servicio ha sido completada de acuerdo a la configuración personalizada del usuario y está lista para ser activada. Ahora, el usuario tiene la capacidad de hacer uso del nuevo servicio, activarlo en el momento que desee, o tener el servicio ejecutándose en segundo plano, en caso de que desee recibir informes constantes sobre los posibles puntos de interés que se relacionan con una tarea específica que el usuario está realizando. Luego, se lleva a cabo una supervisión constante del contexto del usuario y las concordancias de sus preferencias. El siguiente esquema de la Figura 11

muestra Flujo de trabajo (del lado del dispositivo) de manera general el cual se explica detalladamente a continuación:

- El primer paso es la seleccionar componentes, en la que el usuario selecciona de la lista de componentes disponibles en el repositorio de componentes, todos los componentes necesarios para construir el servicio que requiere. Luego, el usuario debe seleccionar los atributos de los componentes seleccionados que le son de interés para las consultas que se realizarán a través del servicio *mashup* que está construyendo.

- El segundo paso es la personalizar el servicio (*mashup)*, y esto incluye la adaptación de las consultas a las preferencias y el nivel de privacidad que ha seleccionado por el usuario. Para lograr esto, se utilizan las preferencias que se corresponden con el ámbito de utilización y los componentes seleccionados para la construcción del servicio. Estas preferencias se almacenan en el dispositivo del usuario y son utilizadas con el fin de comenzar a construir las consultas al servidor.

- El tercer paso es crear y activar el servicio, que se basa en la formulación de las consultas que deben realizarse al servidor y los filtros que se aplicarán en el dispositivo del usuario. La consulta al servidor se basa en los atributos, componentes seleccionados, y las preferencias del usuario que se deben corresponder a cada uno de los atributos. Al mismo tiempo, se considera el nivel de abstracción que el usuario ha seleccionado en materia de presentación de información.

Fig. 11 – Flujo de Trabajo (dispositivo) de creación de servicios personalizados utilizados en PLASMA [73]

El proceso de personalización de los servicios *mashups* sensible al contexto con PLASMA se puede describir más detalladamente en tres etapas (Figura 12). En cada etapa el *mashup* se personaliza cada vez más. Se describen a continuación las tres etapas involucradas:

- La primera etapa se basa en el *Scraping* de diversas páginas web relacionadas con la actividad asociada con un servicio particular. Tiene como objetivo crear un *mashup* con mucho detalle que contenga toda la información posiblemente útil para una experiencia de servicio satisfactoria. Este proceso se realiza en el servidor de componentes (ver Arquitectura en la sección siguiente). Todos los servicios *mashups* creados a partir de esta plataforma se enriquecen con dos tipos adicionales de atributos: La ubicación y las preferencias, con el mayor nivel de precisión posible. Estos dos atributos constituyen el núcleo del proceso de personalización en las etapas subsiguientes. El resultado de esta primera etapa es la creación de un "*mashup*" muy general que sirve de base para la prestación del servicio para cualquier usuario de la plataforma. El *mashup* creado hasta aquí posee mecanismos

de actualización dinámica y se mantiene información válida a través de un "robot".

- La segunda etapa es la parte donde comienza la personalización, a través de la cual se transforma el *"mashup"* original en *"my_mashup"* es decir la versión del *mashup* personalizada. Cada usuario tiene diferentes necesidades y dependiendo del tipo de servicio un usuario puede estar muy interesado en encontrar un resultado deseado en las cercanías, sacrificando así la privacidad de ubicación mientras se respeta la intimidad preferencias personales; mientras que en otros casos es la privacidad de su ubicación donde debe hacerse hincapié. En el servidor existe un espacio personal para cada usuario. El usuario móvil notifica al servidor acerca de ciertos ajustes personales de información, que no son considerados absolutamente privados y que por lo tanto no tiene ningún problema para revelar al servidor. Esta información, se utiliza con el fin de adaptar al mashup de la primera etapa, mashup un uno más personalizado, que es llamado *"my_public_mashup"*, la primera versión pública de *"my_mashup"*. El contenido de *"my_public_mashup"* se envía dispositivo móvil del usuario, como respuesta a las consultas realizadas a un servicio particular.

- La tercera etapa tiene lugar en el dispositivo del usuario. En esta fase los ajustes de personalización que el usuario considera información privada (detalles acerca de las preferencias y la ubicación) y no quiere revelar a terceros se aplican a los contenidos recibidos de *"my_public_mashup"*, y se realiza un nuevo filtrado. El resultado de este filtrado es por tanto una versión totalmente personalizada del mashup, adaptado a las necesidades y preferencias del usuario para la situación concreta. El resultado de la tercera etapa es, *"my_private_mashup"*, que es la segunda versión de

"my_mashup". El contenido de *"my_private_mashup"* es el que se mostrará en la pantalla del dispositivo móvil del usuario.

Fig. 12 – Esquema de detallado del proceso de personalización de *mashups* de PLASMA [74]

4.2.2. Componentes de la Plataforma

En esta plataforma los servicios en lugar de ofrecerse de manera tradicional, están basados en una arquitectura abierta que permite a los usuarios designar, probar, desplegar y compartir componentes de servicios. El marco de trabajo permite el despliegue de servicios que pueden ser personalizados en cuanto al nivel de la privacidad de información personal y exactitud de los datos ofrecidos. Esto depende de la implementación de componentes de servicio que se almacenan en el servidor central y que se puedan acceder a través del dispositivo móvil. Estos componentes de servicio, constituyen el núcleo del servicio, y se basan en *mashups* creados a partir de diversas fuentes, tales como guías turísticos en línea de una ciudad, o bases de datos de películas o música en Internet, periódicos, páginas web estatales, tiendas online, sitios web cartográficos, etc. Además, el propio usuario participa en la creación del servicio *mashup* ya que en el influyen sus preferencias, la configuración de privacidad, las opciones y los comentarios, por lo tanto puede personalizarlo

según sus necesidades. En la Figura 13 se puede ver la arquitectura a nivel general.

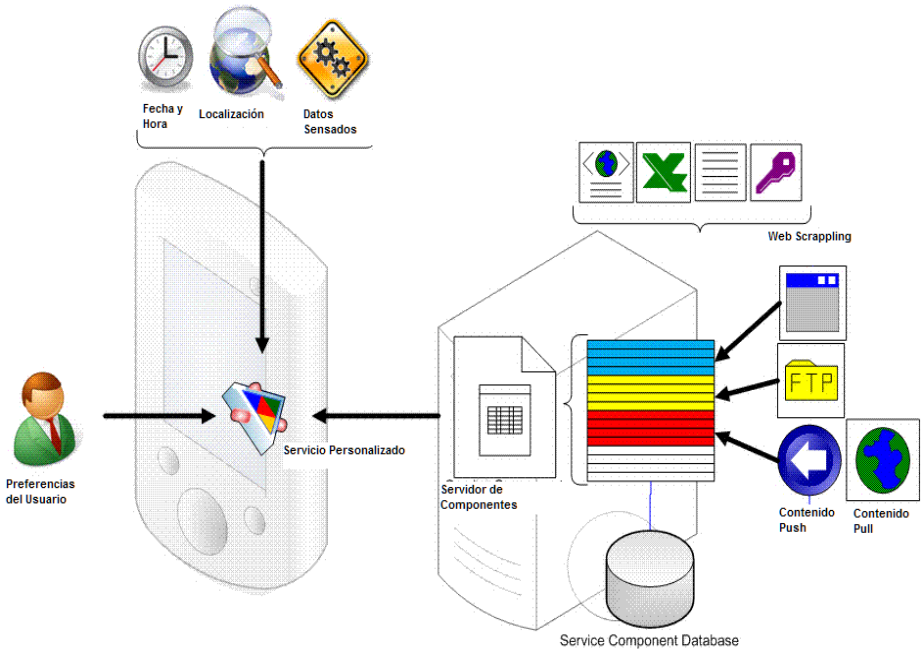


Fig. 13 – Componentes de PLASMA. [74]

4.2.3. Implementación

Para la implementación del cliente PLASMA, se utilizan dispositivos del tipo PocketPC con capacidad de teléfono móvil. La aplicación cliente está construida sobre la plataforma Windows Mobile de Microsoft y se ha utilizado Visual Studio.NET como plataforma de desarrollo. En la Figura. 14 se puede ver una captura de pantalla del cliente ejecutándose en un Smartphone.

Fig. 14 – Pantalla del cliente en un Windows Mobile SmartPhone. [73]

En cuanto al servidor sigue un típico esquema de tres capas, que consta de la capa de Base de datos, la capa de Aplicación y la capa de Presentación. La capa de Base de datos es responsable del mantenimiento de la base de datos en la que se almacenan los componentes de servicio junto con información sobre los puntos de interés que se utilizan para la construcción de los servicios. La base de datos también contiene los procedimientos almacenados necesarios para apoyar los procedimientos de creación de servicios y las consultas del usuario al sistema.

- La capa de Aplicación es la responsable de aplicar la lógica para la interfaz del servicio y la base de datos con el fin de recuperar los resultados de las consultas del usuario.

- La capa de Presentación es responsable de dar soporte a las comunicaciones del servidor con el dispositivo del usuario. Por esta razón se utilizan servicios Web XML.

La base datos esta implementada en MS-SQL Server y en cuanto a los servicios web, estos fueron implementados a través del uso del servidor web Internet Information Server ambos de Microsoft.

4.3. El Caso del Proyecto TELAR

En este trabajo se examina la creación de *mashups* sensibles al contexto, los requisitos que plantean los autores de TELAR para la plataforma en [75] y que luego amplían en [76] se pueden resumir de la siguiente manera:

- La adaptación debe estar basada en sensores que se integran en el dispositivo móvil o conectado localmente.

- Los *mashups* deben ser centrados en el usuario, es decir, el usuario de un dispositivo móvil debe beneficiarse de las aplicaciones web híbridas, en lugar de otros usuarios distantes o los proveedores de servicios.

- Los *mashups* deben ser visibles con el navegador web del dispositivo móvil. Esto impide una solución nativa para el dispositivo móvil y tiene influencia potencial en el rendimiento.

- Debe existir una versión no sensible al contexto del *mashup* en caso de que la información de contexto no esté disponible. Esto permite que se puedan ver los *mashups* en los dispositivos móviles equipados con sensores, así como en equipos que no los posean.

- Debe existir la posibilidad de que múltiples fuentes de datos que mediante formatos de datos distintos e interfaces arbitrarias puedan integrarse en los *mashups*. El usuario debe ser capaz de añadir y eliminar las fuentes de datos en tiempo de ejecución, de acuerdo con su interés actual.

4.3.1. Funcionamiento

La plataforma para el desarrollo TELAR, facilita la creación de *mashups* de localización para dispositivos móviles.

La base para la integración es un mapa provisto por un servicio de mapas en línea (en este caso Google Maps) y de la ubicación actual del usuario, que obtenida mediante de un sensor GPS. Se integran diferentes proveedores de puntos de de interés (POIs) como ser puntos de acceso WLAN de FON, fotos de Panoramio, y los artículos de Wikipedia con referencias geográficas que ofrece Geonombres. Los POIs se muestran en el mapa con diferentes símbolos. A medida que el usuario se mueve, el mapa se mantiene centrado en su posición y se muestra un rastro de su movimiento si el usuario así lo desea. El usuario también puede agregar o quitar información de los POIs que se encuentran en los proveedores durante el tiempo de ejecución. Esto se da a través de los wrappers en el lado del servidor que permiten la integración de los datos de servicios web. Por el lado del cliente, la especificación Delivery Context Client Interfaces (DCCI) [77] se utiliza para integrar la información de contexto de local obtenida a través de sensores con el navegador web móvil, que adapta el *mashup* a la localización actual del usuario.

4.3.2. Componentes de la Plataforma

En la Figura 15 se pueden observar los componentes del sistema TELAR, que de la misma manera que una aplicación *mashup* típica basa en AJAX tiene tres niveles (como se explico en Capítulo 3). El *mashup* se visualiza en la capa del cliente. El navegador carga la página web del *mashup* y ejecuta el código Javascript (AJAX) en el cliente. Esta página del *mashup* se carga desde el servidor de *mashup*, que se encuentra en Internet en la capa servidor. Los datos ofrecidos por terceros proveedores que se utilizan se distribuyen por toda la web, estos se encuentran en la Capa de Proveedores de datos. Los mapas se cargan desde un servicio de mapas, que junto con los proveedores de datos, constituyen la capa de datos. Esta última capa se encuentra fuera de los límites de la organización del *mashup*.

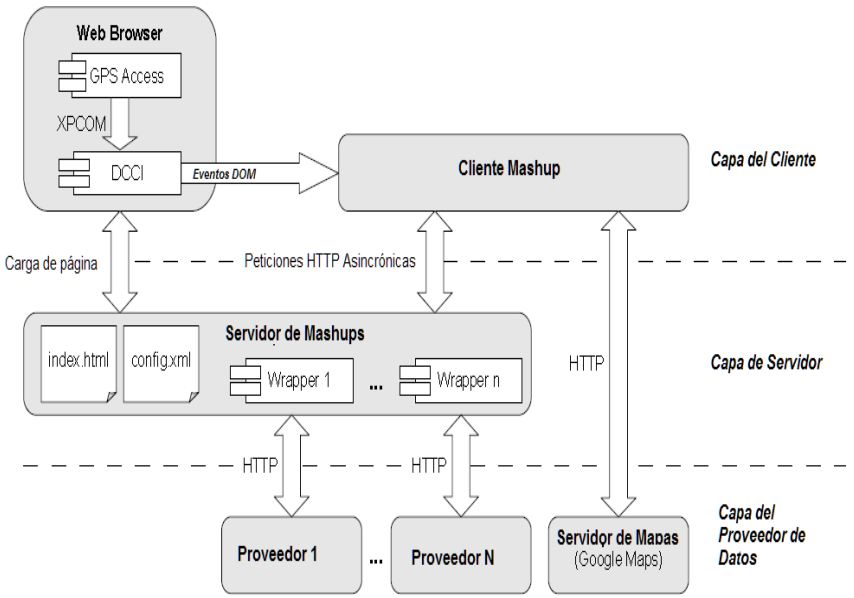


Fig. 15 – Componentes del Proyecto TELAR. [74]

El cliente *mashup* utiliza lo desplegado en el servidor de *mashup* y estos pueden ser configurados/ parametrizados (ej: proveedores de datos a utilizar, posición inicial del mapa, etc.) pero no necesitan ser programados (en algún lenguaje como Java o C++) por los usuarios finales. El cliente *mashup* lee asincrónicamente la configuración cuando la página del *mashup* se carga. Luego el cliente *mashup* construye la interface de usuario, esta muestra el mapa y visualiza los datos POIs de los proveedores de datos.

Con el fin de hacer frente a la heterogeneidad de formatos de datos e interfaces utilizadas por proveedores diferentes, se plantea como necesaria una capa de normalización. Para lograr una interfaz consistente para la recuperación de datos se han utilizados *wrappers*, ya que este enfoque ofrece el más alto grado de flexibilidad. Para otros escenarios con menos diversidad, posiblemente se puedan realizar mediante enfoques de normalización automática (por ejemplo, proveedores de datos que proporcionan canales RSS, como el usado en Yahoo Pipes [61]).

Tanto la información de contexto, como la ubicación del usuario, se integran al *mashup*, ampliando el navegador web. Para ello se utilizan dos componentes adicionales: el módulo DCCI y el módulo de acceso GPS. El módulo que implementa la especificación DCCI constituye la interfaz para proporcionar datos de contexto a las páginas web. El cliente *mashup* se registra como un detector de eventos en el módulo DCCI y este es notificado cada vez que se produce un cambio en la localización del usuario. El módulo de acceso GPS se conecta al dispositivo GPS y este pasa la información de ubicación necesaria al módulo DCCI. Con la separación que brinda el marco de trabajo de adquisición de contexto, en por un lado la interface con el cliente y por otro la adquisición de contexto propiamente dicho, permite que otros módulos de aprovisionamiento de contexto se puedan agregar más adelante.

El flujo de datos funciona de la siguiente manera: Cada vez que el módulo de acceso GPS obtiene una nueva ubicación del dispositivo GPS, la información de ubicación se actualiza en el módulo de DCCI. El cliente *mashup*, que se coloca como un detector de eventos al módulo DCCI, se notifica el cambio a través de eventos DOM. Posteriormente, el cliente actualiza el *mashup* con la nueva ubicación del usuario en el mapa y el mapa se centra en la nueva ubicación. Si el área que se muestra en el mapa ha cambiado significativamente, el cliente envía peticiones HTTP de manera asincrónica a los contenedores, a fin de obtener datos para el área del nuevo mapa. Los *wappers* traducen estas solicitudes en llamadas a la API particular de cada uno de los proveedores de datos y convierte los datos resultantes en un formato de datos único entendido por el cliente *mashup*. Por último, el cliente *mashup* lee la respuesta enviada por *las wappers* y visualiza los datos del POI en el mapa.

4.3.3. Implementación

Para la implementación de la arquitectura los autores han utilizado como dispositivo cliente un Nokia N810 Internet Tablet, como se muestra en la captura de pantalla en la Figura 16.

Fig. 16 – Captura de pantalla de una ejemplo de *Mashup* TELAR en el Nokia N810 Internet Tablet.

El navegador web basado en Mozilla del Nokia N810 ofrece un mecanismo de extensión de gran alcance, mediante la cual el módulo DCCI y el módulo GPS de acceso se desarrollaron en lenguaje C++. Ambos módulos pueden comunicarse directamente a través de XPCOM (Cross Platform Component Object Model), que es un marco de trabajo para el desarrollo de componentes del navegador Mozilla. Como DCCI sólo define una interfaz de cliente, el módulo DCCI tiene que definir su propio interfaz con el proveedor en el back-end. Aunque este trabajo no tiene como objetivo la implementación de un marco de trabajo profesional, esta interfaz de proveedor es genérica y puede ser utilizada para diferentes tipos de proveedores de datos de contexto. Basado en la interfaz del proveedor, se pueden desarrollar más extensiones para el navegador que brinden información de contexto a través de una o más propiedades DCCI.

Para implementar el cliente se utilizó el marco de trabajo Google Web toolkit. Este dio la posibilidad de desarrollar un cliente *mashup* complejo en Java utilizando herramientas profesionales. Por otra parte, los menús y cuadros

de diálogo que componen el interfaz de usuario del cliente *mashup* pueden ser fácilmente diseñadas con la librería de *widgets* del marco de trabajo de Google.

En el caso del lado del servidor, TELAR requiere una aplicación servidor desde la cual el *mashup* se cargue y residan los *wrappers* de los proveedores de datos.

Los *wrappers* se construyeron en PHP 5 (se requiere la extensión JSON y XLS), utilizando hojas de estilo XSL para convertir de formatos de datos basados en XML. Estos son pequeños y no contienen más de 100 líneas de código. Por lo tanto, los proveedores de datos tradicionales se puede agregar a la plataforma de *mashup* propuesta sin problemas y cambios en la interfaces de los proveedores de datos pueden ser fácilmente resueltos.

5. Discusión y Líneas Futuras

Si bien existen herramientas para construir *mashups* sensibles al contexto en entornos de AmI estas son incipientes y todavía están en fase de prototipo. Aún no se cuenta con herramientas con la potencia y facilidad de uso de Yahoo Pipes para escenarios AmI.

Las tecnologías utilizadas, la forma de interacción con los usuarios, y de adquisición de conocimiento de este nuevo tipo de *mashup*, no están estandarizadas y existen múltiples estrategias y plataformas (como las mencionadas en la sección 4) para su construcción por lo que la mayoría de estas propuestas planteen una solución práctica para el escenario específico en cual son aplicadas.

Esto hace difícil integrar datos obtenidos desde sensores y utilizarlos para adaptar el contenido o el comportamiento de los *mashups* (y de cualquier aplicación web) ya que no existe un estándar de cómo capturar, representar y utilizar la información de contexto en las aplicaciones web.

Lo comentado anteriormente da lugar la necesidad de patrones, tanto para la adquisición de datos desde sensores (e información derivada) como en las estrategias para que estos datos sean aprovechados por los sistemas *mashups* de forma segura y que no comprometan la privacidad.

Como líneas futuras se proponen tres pilares, uno desde el punto de vista de las aplicaciones para la creación de *mashups* sensibles al contexto, se plantea la necesidad de contar con una herramienta del tipo de Yahoo Pipes, pero que además de fuentes de datos RSS, APIS, y cajas de texto de entrada de datos (user input) pueda incorporar información de contexto provista por los sensores (context input) del dispositivo móvil del usuario de manera automática.

Siguiendo con esta línea como el segundo pilar sería interesante contar con un navegador web sensible al contexto, que de soporte a la AmI. Este navegador de web ubicua (navegador sensible al contexto), se lo podría plantear como una evolución de un navegador tradicional como Mozilla Firefox o una extensión para los ya existentes. De esta forma se podría realizar una

navegación de Ambientes Inteligentes. Esta extensión o mejora permitiría recabar datos de los sensores de dispositivos móviles de forma transparente para el usuario y las aplicaciones *mashups* e independientemente del hardware o tecnología subyacente.

Como tercer pilar con HTML5 y con formatos interoperables sería interesante abstraer a las aplicaciones *mashups* de las WSN datos a través de un middleware WSN-IP-WWW [78] que ayude al desarrollo de *mashups*.

Bibliografía

[1] Weiser M, The Computer for the 21st Century. Scientific American, 265(3):66–75, September 1991

[2] Weiser, M.. Some computer science issues in Ubiquitous Computing. Communications of ACM, 36(7), 75-84 1993

[3] Sosa, Eduardo O. Contribuciones al establecimiento de una red global de Sensores Inalámbricos. Tesis Doctoral. s.l.: Universidad Nacional de La Plata, Junio 17, 2011.

[4] Information Society Technologies Advisory Group, Orientations for WP2000 and beyond, Sep 1999, ftp://ftp.cordis.europa.eu/pub/ist/docs/istag-99-final.pdf

[5] Noelia Carretero y Ana Belén Bermejo, INTELIGENCIA AMBIENTAL, Centro de Difusión de Tecnologías, Universidad Politécnica de Madrid. www.ceditec.etsit.upm.es/index.php/descargar-documento/5-resumen-inteligencia-ambiental.html

[6] D. Preuveneers, J. Van den Bergh, D. Wagelaar, A. Georges, P. Rigole, T. Clerckx, Y. Berbers, K. Coninx, V. Jonckers, and K. De Bosschere, Towards an Extensible Context Ontology for Ambient Intelligence, 2004

[7] Sitio de Internet: Philips Research Homepage. www.research.philips.com

[8] Sitio de Internet: MIT Project Oxygen Homepage. www.oxygen.lcs.mit.edu

[9] Miguel A. Perez, Loreto Susperregi, Inaki Maurtua, Aitor Ibarguren, Basilio Sierra, "Software Agents for Ambient Intelligence based Manufacturing," dis, pp.139-144, IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS'06), 2006

[10] Sergio A. Velastin, Boghos A. Boghossian, B. P. L. Lo, Jie Sun, M. A. Vicencio-Silva: PRISMATICA: toward ambient intelligence in public

transport environments. IEEE Transactions on Systems, Man, and Cybernetics, Part A 35(1): 164-182 (2005)

[11] O. Stock, M. Zancanaro, P. Busetta, C. Callaway, A. Krüger, M. Kruppa, T. Kuflik, E. Not, and C. Rocchi, "Adaptive, intelligent presentation of information for the museum visitor in peach," User Modeling and User-Adapted Interaction, vol. 17, no. 3, pp. 257-304, July 2007

[12] Andreas Bulling , Daniel Roggen , Gerhard Tröster, Wearable EOG goggles: Seamless sensing and context-awareness in everyday environments, Journal of Ambient Intelligence and Smart Environments, v.1 n.2, p.157-171, April 2009

[13] Sitio de Internet: Project COSM http://www.COSM.com/

[14] Schilit, B., Theimer, M. Disseminating Active Map Information to Mobile Hosts. IEEE Network, 8(5) (1994) 22-32

[15] Brown, P.J., Bovey, J.D. Chen, X. Context-Aware Applications: From the Laboratory to the Mar-ketplace. IEEE Personal Communications, 4(5) (1997) 58-64

[16] Ryan, N., Pascoe, J., Morse, D. Enhanced Reality Fieldwork: the Context-Aware Archaeological Assistant. Gaffney,V., van Leusen, M., Exxon, S. (eds.) Computer Applications in Archaeology (1997)

[17] Dey, A.K. Context-Aware Computing: The CyberDesk Project. AAAI 1998 Spring Symposium on Intelligent Environments, Technical Report SS-98-02 (1998) 51-54

[18] Brown, P.J. The Stick-e Document: a Framework for Creating Context-Aware Applications. Electronic Publishing ’96 (1996) 259-272

[19] Franklin, D., Flaschbart, J. All Gadget and No Representation Makes Jack a Dull Environment. AAAI 1998 Spring Symposium on Intelligent Environments, Technical Report SS-98-02 (1998) 155-160

[20] Ward, A., Jones, A., Hopper, A. A New Location Technique for the Active Office. IEEE Personal Communications 4(5) (1997) 42-47

[21] Schilit, B., Adams, N. Want, R. Context-Aware Computing Applications. 1st International Workshop on Mobile Computing Systems and Applications. (1994) 85-90

[22] Dey, A.K., Abowd, G.D., Wood, A. CyberDesk: A Framework for Providing Self-Integrating Con-text-Aware Services. Knowledge-Based Systems, 11 (1999) 3-13

[23] Pascoe, J. Adding Generic Contextual Capabilities to Wearable Computers. 2nd International Sympo-sium on Wearable Computers (1998) 92-99

[24] Anind Kumar Dey. Providing architectural support for building context-aware applications. PhD thesis, Georgia Institute of Technology, 2000. Director-Gregory D. Abowd.

[25] Gerd Kortuem, Zary Segall, and Martin Bauer. Context-aware, adaptive wearable computers as remote interfaces to 'intelligent' environments. In ISWC, pages 58–65, 1998.

[26] Salber, D., Dey, A.K., Abowd, G.D. Ubiquitous Computing: Defining an HCI Research Agenda for an Emerging Interaction Paradigm. Georgia Tech GVU Technical Report GIT-GVU-98-01 (1998)

[27] A.K. Dey. Understanding and using context. In Personal and Ubiquitous Computing Journal, volume 5, pages 4-7, 2001.

[28] H. Chen, T, Finin, A. Joshi. "Semantic Web in a Pervasive Context-Aware Architecture", University of Maryland. Artificial Intelligence in Mobile Systems.2003.

[29] D. Fournier, S. Mokhtar, N. Georgantas, V. Issarny. "Towards Ad hoc Contextual Services for Pervasive Computing". In Proceedings of the

International Workshop on Middleware for Service Oriented Computing. 2006.

[30] M. Baldauf y S. Dustdar. “A Survey on Context-Aware Systems”. International Journal of Ad Hoc and Ubiquitous Computing. 2004.

[31] Caus, T., Christmann, S, Hagenhoff, S. 2008. Hydra - An Application Framework for the Development of Context-Aware Mobile Services, in: W. Abramowicz, D. Fensel (Eds.), 11th International Conference on Business Information Systems, Innsbruck, Austria, 2008, pp. 471-482.

[32] A. Held, S. Buchholz and A. Schill. “Modeling of context information for pervasive computing applications”. In Proceedings of SCI. 2002

[33] J. Bauer. “Indentification and Modeling of Contexts for Different Information Scenarios in Air Traffic”. Diplomarbeit. 2003

[34] K. Henricksen, J. Indulska and A. Rakotonirainy. “Generating Context Management Infrastructure from HighLevel Context Models”. In Industrial Track Proceedings of the 4th International Conference on Mobile Data Management (MDM 2003), pp. 1-6. 2003.

[35] A. Schmidt and K. V. Laerhoven. “How to Build Smart Appliances”. IEEE Personal Communications. 2001.

[36] P. Gray and D. Salber. “Modelling and Using Sensed Context Information in the design of Interactive Applications”. Proceedings of 8th IFIP International Conference on Engineering for Human Computer Interaciton. 2001.

[37] H. Chen, T. Finin and A. Joshi. “Using OWL in a Pervasive Computing Broker”. In Proceedings of Workshop on Ontologies in Open Agent Systems. 2003.

[38] H. Chen, T. Finin and A. Joshi. An Ontology for Context-Aware Pervasive Computing Environments”. Special Issue on Ontologies for Distributed Systems, pp. 197-207. Cambridge University Press. 2004.

[39] T. O'Reilly, "What Is Web 2.0, Design Patterns and Business Models for the Next Generation of Software", O'Reilly Media Inc., September 2005. http://oreilly.com/web2/archive/what-is-web-20.html

[40] Pasado, presente y futuro de la Web 2.0 http://www.ub.edu/bid/17serra2.htm

[41] Koschmider, A., Torres, V., Pelechano, V., Elucidating the mashup hype: Definition, Challenges,Methodical guide and tools for mashups. In: 2nd Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web. 2009.

[42] Zou, J.; Pavlovski, C.J., "Towards Accountable Enterprise Mashup Services," e-Business Engineering, 2007. ICEBE 2007. IEEE International Conference on , vol., no., pp.205-212, 24-26 Oct. 2007

[43] Duane Merrill. Mashups: The new breed of Web app--Anintroduction to mashups. http://www.ibm.com/developerworks/xml/library/x-mashups.html

[44] Xuanzhe Liu; Yi Hui; Wei Sun; Haiqi Liang, "Towards Service Composition Based on Mashup," Services, 2007 IEEE Congress on , vol., no., pp.332-339, 9-13 July 2007

[45] Sitio de Internet: Screen Scrapper, http://www.screen-scraper.com/

[46] Hoyer, V., and Fischer, M.: Market Overview of Enterprise Mashup Tools. In International Conference on Service oriented Computing, Volume 5364 of Lecture Notes in Computer Science, Springer-Verlag, 2008, pp. 708-721.

[47] Sitio de Internet: Programmableweb – mashups,: http://www.programmableweb.com/mashups

[48] Li, S., and Gong, J.: Mashup: a New Way of Providing Web Mapping and GIS Services. In ISPRS Congress Beijing 2008, Proceedings of Commission IV, 2008, pp. 639-649.

[49] Sitio de Internet: Dion Hinchcliffe, Is IBM making enterprise mashups respectable ZDNet Blog, 2006. http://blogs.zdnet.com/Hinchcliffe/?p=49&tag=nl.e622

[50] Hartmann, B. and Doorley, S. and Klemmer, S. R.: Hacking, Mashing, Gluing: Understanding Opportunistic Design, In IEEE Pervasive Computing, IEEE Educational Activities Department, 7(3), 2008, pp. 1536-1268

[51] Sitio de Internet: W3C Working Group Note 11, (2004) Web Services Architecture, W3C Working Group Note, disponible en: http://www.w3.org/TR/ws-arch/

[52] Sitio de Internet: W3C Consortium, Web Services Description Working Group, disponible en: http://www.w3.org/2002/ws/desc/

[53] Fielding, R., Architectural Styles and the Design of Network-based Software, University of California, 2000.

[54] W3C Consortium , W3C Recommendation, (2007), SOAP Version 1.2 Part 1: Messaging Framework (Second Edition), http://www.w3.org/TR/soap12-part1/

[55] RDF Site Summary (RSS) 1.0, disponible en: http://web.resource.org/rss/1.0/spec

[56] RSS 2.0 Specification, disponible en: http://validator.w3.org/feed/docs/rss2.html

[57] RFC 4287, (2005), The Atom Syndication Format December 2005. http://www.ietf.org/rfc/rfc4287.txt

[58] Sitio de Internet: Ajax Tutorial, disponible en: http://www.w3schools.com/Ajax/Default.Asp

[59] Sitio de Internet: Screen Scrapper, http://www.screen-scraper.com/

[60] Sitio de Internet: W3C Consortium , Resource Description Framework, http://www.w3.org/RD/

[61] Sitio de Internet: Yahoo Pipes, http://pipes.yahoo.com/pipes/ Visita:

[62] Sitio de Internet: Deri Pipes: http://pipes.deri.org/

[63] Sitio de Internet: SnapLogic Integration Platform, http://www.snaplogic.com

[64] Hinchcliffe, D., (2006), Assembing Great Software: A Round-Up of Eight Mashup Tools, disponible en: http://blogs.zdnet.com/Hinchcliffe/?p=63

[65] Sitio de Internet: WSO2 Mashup Server, http://wso2.org/projects/mashup

[66] Sitio de Internet: Open Mashups Studio, http://www.open-mashups.org/

[67] Godoy, D.; Sosa, E. Díaz Redondo, R. ;"Plataformas para la Creación de Mashups en Entornos de AmI" en las VII Jornadas científico tecnológicas de la Universidad Nacional de Misiones. Página 163. ISBN 978-950-766-081-8.

[68] Godoy, D; Sosa, E. "Mashups En Ambientes Inteligentes" XV Workshop de Investigadores en Ciencias de la Computación 2013. Universidad Autónoma de Entre Ríos. Páginas 963-972 ISBN 978-987-281-796-1

[69] D. Lopez-De-Ipina, J. I. Vazquez, J. Abaitua. "A context-aware mobile mashup platform for ubiquitous web". Intelligent Environments, 2007. IE 07. 3rd IET International Conference on (2007), pp. 116-123.

[70] López-de-Ipiña D, Vazquez JI, Abaitua J, A Web 2.0 platform to enable context-aware mobile mashups. In: Proceedings of AmI'2007 on ambient intelligence, Darmstadt, Germany, 7–10 November 2007, pp 266–286

[71] López de Ipiña D., Vázquez J.I. and Abaitua J., Context-Aware Mobile Mash-up for Ubiquitous Web, Proceedings of 2nd International Workshop on Ubiquitous Computing and Ambient Intelligence, Puertollano, Spain, ISBN: 84-6901744-6, pp. 19-34, November 2006

[72] Sitio de Internet: W3C, Synchronized Multimedia, SMIL Home Site, http://www.w3.org/AudioVideo/ , 2010.

[73] A. Voulodimos and C. Patrikakis. Using personalized mashups for mobile location based services. Wireless Communications and Mobile Computing Conference, pages 321 – 325, 2008

[74] Charalampos Patrikakis, Athanasios Voulodimos , George Giannoulis, Personalized location based services with respect to privacy: a user oriented approach, Proceedings of the 2nd International Conference on PErvsive Technologies Related to Assistive Environments, p.1-5, June 09-13, 2009, Corfu, Greece.

[75] Andreas Brodt , Daniela Nicklas, The TELAR mobile mashup platform for Nokia internet tablets, Proceedings of the 11th international conference on Extending database technology: Advances in database technology, March 25-29, 2008, Nantes, France

[76] A. Brodt, D. Nicklas, S. Sathish, and B. Mitschang. Contextaware mashups for mobile devices. In Web Engineering (WISE ’08), pages 280–291. Springer-Verlag, 2008.

[77] K. Waters, R. A. Hosn, D. Raggett, S. Sathish, M. Womer, M. Froumentin, and R. Lewis. Delivery Context: Client Interfaces (DCCI) 1.0. Working draft, W3C, July 2007. http://www.w3.org/TR/2007/WD-DPF-20070704/

[78] Benitez, J; Gloza, G; Sosa, E.Godoy, D “Conectividad WSN: Implementación de un Middleware WSN-IP-WWW” XV Workshop de Investigadores en Ciencias de la Computación 2013. Universidad Autónoma de Entre Ríos. Páginas 28-33 ISBN 978-987-281-796-1

Publicaciones


VIII JORNADAS
Científico
Tecnológicas
Facultad de Cs. Exactas, Químicas y Naturales
Universidad Nacional de Misiones
2, 3 y 4 de noviembre de 2011
Museo Provincial de Bellas Artes "Juan Yaparí"
Posadas, Misiones, Argentina | www.fceqyn.unam.edu.ar | jornadassecip2011@gmail.com
Año Internacional de la
QUÍMICA
2011

VIII Jornadas de Investigación Científico Tecnológicos de la Facultad de Ciencias Exactas, Químicas y Naturales de la Universidad Nacional de Misiones: 2, 3 y 4 de noviembre de 2011 [CD-ROM] / Laura Lidia Villalba...[et.al.]. 1a ed. Posadas: Universidad Nacional de Misiones, 2011.

ISBN: 978-950-766-081-8

1. Enseñanza Universitaria, Investigación. I. Villalba, Laura Lidia
CDU 001.891

Fecha de catalogación: 27/10/2011

I.07
PLATAFORMAS PARA LA CREACIÓN DE MASHUPS EN ENTORNOS DE AMI

Godoy, Diego Alberto[1,a], Sosa, Eduardo Omar[2,b], Díaz Redondo, Rebeca[3,c]

[1]Universidad Gastón Dachary, [2]Facultad de C. Exactas, Químicas y Naturales. UNaM.
[3]Universidad de Vigo, E.T.S.I. Telecomunicación
correo electrónico: [a]diegodoy@ugd.edu.ar, [b]es@fceqyn.unam.edu.ar, [c]rebeca@det.uvigo.es

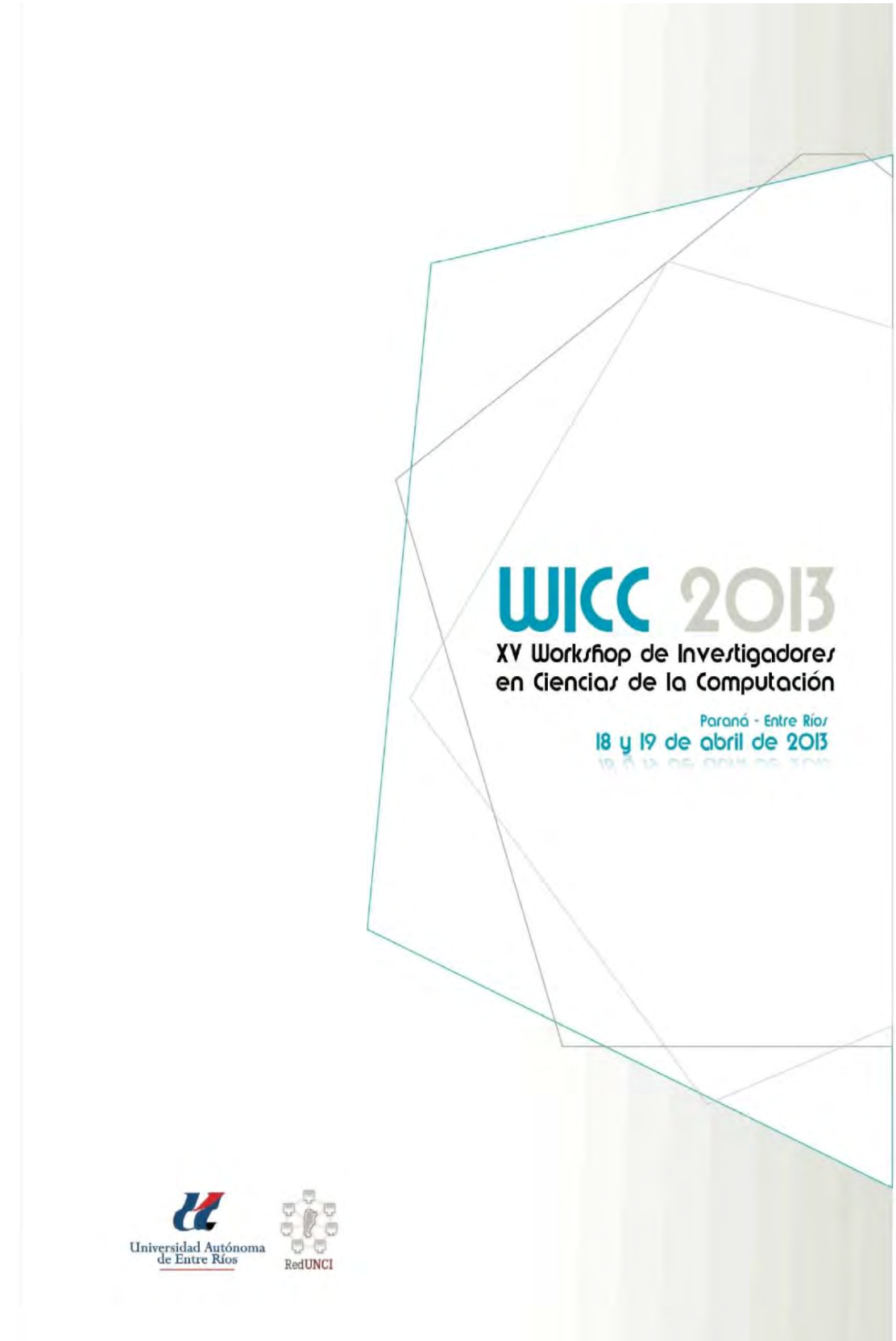
Palabras Claves: Mashups, Inteligencia Ambiental, IoT, Sensibilidad al Contexto, WSN

En el presente trabajo se investigaron tres plataformas existentes para la creación de mashups sensibles al contexto en entornos de Inteligencia Ambiental (AmI), para establecer el estado del arte y proponer algunas alternativas de mejora. En los últimos años se ha observado el avance tanto de las tecnologías envueltas en la denominada **"Internet de las Cosas (IoT)" representadas fundamentalmente por** las Redes de Sensores Inalámbricos (WSN) y los dispositivos RFID, como también de las tecnologías implementadas a nivel aplicación en la WWW. Todo ello basado en protocolos más eficientes, mayores anchos de banda, dispositivos de tamaños microscópicos, mayores áreas de cobertura, menor consumo de energía y nuevos tipos de sensores; conformando una "sociedad de redes ubicuas". Sumado a ello el desarrollo y consecuente requerimiento de nuevos servicios on-line precisa de una nueva interfaz de interacción aplicación-usuario. Esto ha provocado una evolución hacia lo que se ha dado en llamar web 2.0. Allí los usuarios son consumidores y generadores de la información publicada, y la experiencia del usuario al usar estos sistemas se hizo más agradable, posibilitando personalizar las aplicaciones a las necesidades concretas. Para estos entornos WWW se han desarrollado **mezclas o "mashups" tanto de información como de comportamiento de aplicaciones**, sin necesidad de un profundo conocimiento de programación en Java o C++. Con ello AmI adhiere al concepto del (Information Society Technologies Program Advisory Group) ISTAG referido a personas rodeadas de interfaces de IoT permitiendo Interacción de forma natural con diferentes sistemas de información.

Las plataformas que han sido analizadas son: 1) El proyecto *Deusto Sentient Graffiti* que busca la convergencia de la computación móvil y la computación ubicua con la web 2.0 para configurar lo que hoy día se llama Web Ubicua (UW). 2) El proyecto *TELAR*, que examina la creación de mashups sensibles al contexto, los cuales deben ser centrados en el usuario y visibles en un navegador de un dispositivo móvil asegurando compatibilidad de formatos. 3) El proyecto *Plasma*, que permite a sus usuarios construir, desplegar y distribuir servicios móviles personalizados, posibilitando la integración de contenido estático y dinámico de varias fuentes de la Web, permitiendo al mashup dar soporte a la provisión de servicios móviles personalizados y cumplir la visión de la AmI.

Se concluye que si bien existen herramientas para construir mashups sensibles al contexto en entornos de AmI, estas son incipientes y todavía están en fase de prototipo. La forma de interacción con los usuarios, y de adquisición de conocimiento de este nuevo tipo de mashups sensibles al contexto, no están estandarizadas y existen múltiples estrategias y plataformas para su construcción por lo que la mayoría de estas propuestas plantean una solución práctica para cada escenario específico. Esto dificulta integrar datos capturados y procesados por una red ubicua al comportamiento de los mashups ya que no existe una especificación al respecto.

A futuro se plantea la necesidad de contar con una herramienta adecuada para la creación de mashups sensibles al contexto similar a Yahoo Pipes, pero que incorpore información de contexto provista por los sensores del dispositivo móvil del usuario o WSNs a través de un gateway utilizando REST. Asimismo es dable de considerar un navegador sensible al contexto, con soporte AmI, ya sea como una evolución de un navegador tradicional o como una extensión; la que permita recabar datos e información de contexto de forma transparente tanto para el usuario y a las aplicaciones.


WICC 2013
XV Workshop de Investigadores en Ciencias de la Computación
Paraná - Entre Ríos
18 y 19 de abril de 2013
Universidad Autónoma de Entre Ríos
RedUNCI

WICC 2013

XV Workshop de Investigadores en Ciencias de la Computación

Paraná - Entre Ríos

18 y 19 de abril de 2013

XV Workshop de Investigadores en Ciencias de la Computación 2013

ISBN: 9789872817961

WICC 2013

El Workshop de Investigadores en Ciencias de la Computación es organizado, a partir de 1999, por la Red de Universidades Nacionales con Carreras de Informática (RedUNCI). El objetivo del Workshop es crear un foro para el intercambio de ideas entre Investigadores en Ciencias de la Computación, de modo de fomentar la vinculación y potenciar el desarrollo coordinado de actividades de Investigación y Desarrollo entre los mismos.

Facultad de Ciencia y Tecnología

Universidad Autónoma de Entre Ríos (UADER)

wicc2013@uader.edu.ar

Compilación:

ASS Claudio Caluva
Lic. Silvia Aranguren
Lic. Rodolfo Muzachiodi

Mashups en Ambientes Inteligentes

Diego A. Godoy[(1,a)];, Eduardo O. Sosa[(1,b)];

[(1)]Centro de Investigación en Tecnologías de la Información y Comunicaciones (C.I.T.I.C.). Universidad Gastón Dachary (U.G.D.)

Campus Urbano, UGD. Posadas, Argentina
Teléfono: +54-376-4438677

[(a)]diegodoy, [(b)]eduardo.sosa@citic.dachary.edu.ar

Resumen

La aparición de los mashups conjuntamente con la Inteligencia Ambiental abre nuevas posibilidades para el desarrollo de aplicaciones que se adapten al contexto de sus usuarios. Esto da lugar a la necesidad de nuevas arquitecturas en las que se posibiliten la creación de mashups que sean sensibles al contexto y que puedan ser creados de forma práctica por el usuario. Es por ello que en este trabajo se analizarán dos arquitecturas existentes para la creación de esta nueva clase de mashups y se discutirán posibles mejoras. El trabajo presentado aquí se realiza en el marco del proyecto de investigación denominado "Diseño de arquitecturas de soporte a la Internet del Futuro y Ambientes Inteligentes" que se ha formalizado por la resolución (19/A/12) de la Universidad Gastón Dachary.

Palabras clave: Mashups, Sensibilidad al Contexto, Inteligencia Ambiental

Contexto

El trabajo presentado aquí se realiza en el marco del proyecto de investigación denominado "Diseño de arquitecturas de soporte a la Internet del Futuro y Ambientes Inteligentes" que se ha formalizado por la resolución (19/A/12) del la U.G.D. mediante el llamado al 5to Concurso de Proyectos de Investigación en 2012. EL objetivo de este proyecto es "Diseñar arquitecturas de soporte a Internet del futuro y Ambientes Inteligentes para su aplicación a Ciudades Inteligentes". Como antecedentes y resultados parciales del mencionado proyecto se pueden indicar 3 proyectos de trabajos de grado en curso, correspondientes a la Carrera de Ingeniería en Informática de la U.G.D. y un trabajo final especialización Ingeniería de Software de la Universidad Nacional de la Plata. Indicar el proyecto en que está inserta la línea de I/D presentada y la/s instituciones que coordinan el proyecto. En los casos que corresponda indicar la Institución que acredita el proyecto y los organismos/empresas que contribuyen a su financiamiento.

Introducción

En los últimos años hemos sido testigos del gran desarrollo de dos tecnologías: los dispositivos móviles (basados en redes inalámbricas, con GPS y sensores) y la Web.

En cuanto a la primera de estas tecnologías se han desarrollado grandes avances en redes inalámbricas, como ser: protocolos más eficientes, mayores anchos de banda, mayores áreas de cobertura, nuevos tipos de sensores, etc. Además los dispositivos móviles cuentan cada vez más, con mayor poder de cómputo, pantallas más grandes y se consiguen a un menor precio. Estos avances facilitan el desarrollo de aplicaciones en donde la movilidad y ubicuidad son parte esencial.

Sin duda una de las características más importantes de los sistemas móviles, es que brindan la posibilidad de aprovechar la posición geográfica para asistir al usuario mientras este se desplaza. Ejemplos de ello son los sistemas que guían a los usuarios en las visitas a museos, sistemas de navegación basados en GPS, o proyectos como Friend Finder de A&T que notifica la presencia de personas geográficamente cercanas.

Hoy en día el desarrollo de sistemas para escenarios móviles ha aumentado considerablemente. Se ha pasado de aplicaciones casi experimentales o utilizadas solo en ambientes académicos, a algunas aplicaciones comerciales como las orientadas a negocios, el uso personal o el ocio.

De la misma manera que las redes inalámbricas, el aumento del poder de cómputo de los dispositivos y capacidad de censado de los dispositivos móviles supone una revolución, el crecimiento de la web también fue revolucionario en cuanto a la forma de presentar la información a los usuarios de sistemas informáticos. Esto dio origen a una evolución de la web, llamada web 2.0, en donde ya los usuarios no son únicamente consumidores, y donde la experiencia del usuario a usar estos sistemas se hizo más agradable, posibilitando personalizar las aplicaciones a las necesidades concretas de los usuarios y mejorar la interfaz gráfica.

Con estas mejoras de la web, muchas veces ya no son suficiente sitios donde se puedan obtener datos de una sola fuente, es más, se hace sumamente necesario contar con datos de diversas fuentes integradas, y para ello se han desarrollado plataformas que permiten hacer mezclas o "mashups" tanto de información como de comportamiento de las aplicaciones. La particularidad de estas mezclas es que son los mismos usuarios los que tienen la posibilidad de crearlas para cumplir con sus requisitos y no necesitan de un programador o conocer lenguajes como Java o C++.

La Inteligencia Ambiental (AmI) o Computación Ubicua no está ajena a los avances de estas dos tecnologías y se ve beneficiada por ambas. Estos avances permiten que la AmI se acerque cada vez más al cumplimento a la visión del Information Society Technologies Program Advisory Group (ISTAG) de personas rodeadas de interfaces inteligentes embebidas en objetos de la vida cotidiana que permitan la interacción de forma natural y sin esfuerzo con diferentes sistemas de información.

Este trabajo tiene como objetivo conocer cuáles son las plataformas existentes para la creación de mashups sensibles al contexto en entornos de AmI y proponer mejoras a estas.

La evolución y creciente miniaturización de los dispositivos de cómputo, que hace posible que pequeños procesadores y diminutos sensores se integren cada vez más en objetos cotidianos. Mark Weiser ya había previsto ésta evolución hace más de 10 años y la publicó en su artículo "The Computer for the 21st Century" [1]. En un trabajo posterior Weiser se planteaba los siguientes interrogantes "¿Cuál es la idea de la computadora del futuro? ¿El agente inteligente? , ¿La televisión (multimedia)?, ¿El mundo gráfico

3-D (realidad virtual)? ¿El equipo de voz ubicua de la película StarTrek?, ¿Las interfaces gráficas de usuarios –GUI-, elaboradas y seleccionadas?, ¿La máquina que por arte de magia cumpla y acceda a nuestros deseos?" [2]. La respuesta que él mismo ha dado es: ninguna de las anteriores. La sencilla razón es que todos estos conceptos comparten un defecto básico: se llevan a la práctica con algo tangible, visible.

Es así Weiser que definió el concepto de "computación ubicua", refiriéndose a las computadoras omnipresentes al servicio de las personas en su vida cotidiana, en el hogar y en el trabajo, y funcionando de manera invisible y no intrusita. Los principios enunciados originalmente por Weiser han sido validados por diferentes investigadores e ingenieros, convirtiéndose en un escenario real en ciertos ámbitos y extendiéndose a escala global [3].

En 1999, el Information Society Technologies Program Advisory Group (ISTAG) de la Unión Europea ha utilizado el término "inteligencia ambiental" del inglés Ambient Intelligence (AmI) de manera similar para describir una visión donde "las personas estarán rodeadas de interfaces inteligentes e intuitivas embebidas en objetos cotidianos de nuestro alrededor y un entorno que reconocen y responden a la presencia de individuos de manera invisible [4]. Es así que el objetivo de la AmI consiste en la creación de espacios donde los usuarios interaccionen de forma natural y sin esfuerzo con los diferentes sistemas.

Uno de los aspectos más importantes de la Inteligencia ambiental son las aplicaciones sensibles al contexto, es por ello que es necesario entender a que nos referimos con este concepto. Existen varias definiciones: La primera fue dada en [5], la cual restringía a estas a ser simplemente aplicaciones que eran informadas sobre el contexto para que se adaptaran a este. Otras definiciones como la de [6] definían a la computación sensible al contexto como la habilidad que poseen los dispositivos de detectar, sensar, interpretar y responder a los aspectos locales al ambiente de un usuario. Por su parte, Dey [7] define la noción de context-awareness como el "uso del contexto para automatizar un sistema de software, modificar su interface y proveer la máxima flexibilidad en términos de servicios".

Otra definición un poco más pasiva es la dada en [8], donde una aplicación sensible al contexto es aquella que pueda variar o adaptar dinámicamente su comportamiento en base al contexto. Salber [9] define que una aplicación sensible al contexto es aquella que tiene la capacidad de proporcionar la máxima flexibilidad de un servicio de cómputo basado en el censado del contexto en tiempo real.

Por último, en una forma más general pero a la vez concreta, en [10] Dey define a un sistema sensible al contexto como: "Un sistema (aplicación) es sensible al contexto si este hace uso de información del contexto para proveer información o servicios relevantes al usuario, donde la relevancia depende de actividad actual del usuario". Esta definición resulta ser la mayormente aceptada.

El auge de la Web 2.0 ha hecho que la industria incorpore las nuevas tecnologías que dieron lugar a su origen y se ajusten a los estándares emergentes. Mientras aun hoy no existe consenso sobre el alcance del término Web 2.0, O'Reilly ofrece una definición comúnmente aceptada, que incluye una gama de servicios mejorados, como ser los servicios web, wikis, blogs, bittorrents, y la sindicación de contenidos [11].

El crecimiento de la Web 2.0 también ha introducido un número de patrones de diseño y nuevos estilos arquitectónicos en el desarrollo

web. Una de las técnicas más destacadas consiste en la "mezcla" o mashup.

En [12] se presenta una definición precisa y completa la cual define a un mashup como una "Aplicación basada en Web que se crea mediante la combinación y procesamiento de los recursos en línea de terceros, que contribuyen con datos, forma de presentación o funcionalidad". En esta definición, los recursos en línea de terceros se refieren a cualquier tipo de recursos disponibles en Internet, independientemente del formato. Como resultado, un mashup proporciona un nuevo recurso.

En términos generales, los mashup se crean usando el navegador web, "arrastrando y soltando" es decir, poniendo juntas aplicaciones de diferentes fuentes. Sin embargo, estas operaciones deben tener una infraestructura en el back-end para que de soporte al mashup, por ello en [13,14], se presenta una arquitectura de mashup que consta de tres componentes la cual se puede ver en la Figura 1. a) La API / proveedores de contenido. Estos son los proveedores de contenido (a veces heterogéneos) de los cuales se desea hacer el mashup. Para facilitar la recuperación de datos, los proveedores suelen dar acceso a sus contenidos a través de protocolos de Internet tales como REST, los Web Services o feeds RSS / Atom o utilizando screen scraping [15].b) El hosting del mashup. Es donde el mashup se encuentra alojado. Los mashups pueden utilizarse de manera similar a las aplicaciones Web tradicionales utilizando un servidor de contenido dinámico (Java Servlets, CGI, PHP o ASP). Alternativamente, el contenido del mashup se puede generar directamente en el navegador del cliente utilizando por ejemplo JavaScript o applets. c) El navegador Web de los consumidores. Es donde la solicitud se representa gráficamente y la interacción del usuario con el mashup se lleva a cabo.

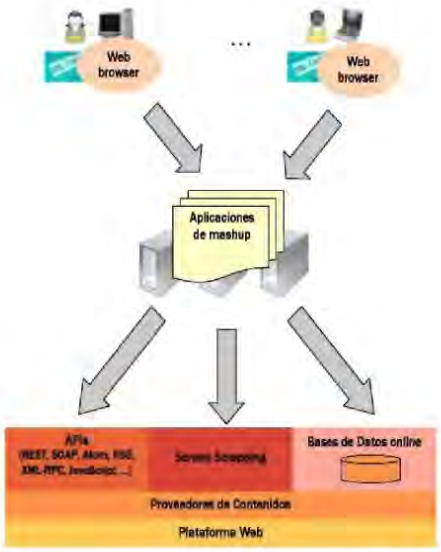


Figura 1. Arquitectura de Mashups

Líneas de investigación y desarrollo

En esta sección se analizarán dos proyectos de plataformas existentes para construir mashups sensibles al contexto:

1) El Caso del Proyecto Deusto Sentient Graffiti (DSG):El proyecto Deusto Sentient Graffiti [16, 17] busca la convergencia de la computación móvil y la computación ubicua con la web 2.0 para configurar lo que hoy día se llama Web Ubicua (UW). Esta última se puede definir como una infraestructura de Internet omnipresente en la que todos los objetos físicos son recursos accesibles a través de una URI. Estas URIs pueden proporcionar información y servicios que enriquezcan la experiencia de los usuarios en su contexto físico.

Su funcionamiento se basa en las anotaciones espontáneas o "Graffitis" (en formato de documento XML) que realiza una comunidad de usuarios sobre objetos, lugares o incluso

personas, con contenido web multimedia y/o servicios que pueden ser descubiertos y utilizados a su vez por otros usuarios móviles cuyos atributos contextuales coincidan con las de las anotaciones. La funcionalidad de DGS consiste en dos procesos principales: 1) Anotación de Grafitis: Los usuarios de los clientes DGS anotan objetos o regiones espaciales las cuales consistiendo de Descripciones, palabras clave y atributos contextuales y 2) descubrimiento y consumo de Grafitis, cuando los usuarios, pueden moverse virtual o fisicamente a través de un ambiente navegando y consumiendo las anotaciones disponibles

DGS puede verse como mashup sensible al contexto dado que los clientes de DGS consiguen combinar la información geográfica en forma de mapas (ejemplo Google Maps) y la información suministrada por el servidor DSG en las descripciones.

La arquitectura de DGS presenta los siguientes componentes (Figura 2): a) Administrador de Grafiti y Web Server: administra todos los grafitis virtuales publicados b) Disparador y Notificador de Grafitis: Es un motor de inferencia basado en reglas que deduce los conjuntos de anotaciones activas que tienen notificarse a los usuarios c) Motor de Consultas de Grafitis: Este componente es responsable del procesamiento de consultas y de la recuperación de la anotaciones sensibles al contexto d) Monitores de Cambio de Contexto: Se ejecutan en el dispositivo móvil o navegador web vigilando el contexto del usuario, y e) Reproductor de Grafitis: Muestra grafitis virtuales en el dispositivo del usuario.

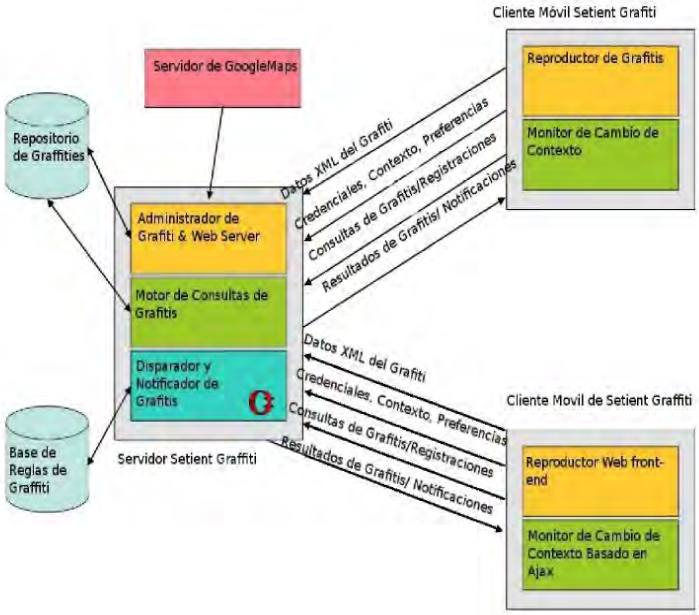


Figura 2. Componentes de DSG

Para su implementación DGS utiliza una sintaxis llamada GraffitiXML que se usa para especificar contenido. Como todo sistema basado en Web 2.0, como Google maps o Flickr, DGS provee una API basada en HTTP que permite a programadores de terceras partes desarrollar aplicaciones que se beneficien de su funcionalidad.

La implementación del servidor DGS está basada en Java y se ejecuta sobre el servidor de aplicaciones Tomcat utilizando JAVA EE. Además utiliza los marcos de trabajo, Tapesty, Hibernate, y Java Expert Systems Shell y como base de datos MySQL.

2) El Caso del Proyecto TELAR: Los autores de TELAR para la plataforma en [18] y que luego amplían en [19] se pueden resumir de la siguiente manera: a) La adaptación debe estar basada en sensores que se integran en el dispositivo móvil o conectado localmente. B) Los mashups deben ser centrados en el usuario, es decir, el usuario de un dispositivo móvil debe beneficiarse de las aplicaciones web híbridas, en lugar de otros usuarios distantes o los proveedores de servicios, c) Los mashups deben ser visibles con el navegador web del dispositivo móvil. Esto impide una solución nativa para el dispositivo móvil y tiene influencia potencial en el rendimiento, d) Debe existir una versión no sensible al contexto del mashup en caso de que la información de contexto no esté disponible. Esto permite que se puedan ver los mashups en los dispositivos móviles equipados con sensores, así como en equipos que no los posean e) Debe existir la posibilidad de que múltiples fuentes de datos con formatos de datos distintos e interfaces arbitrarias puedan integrarse en los mashups.

La base para la integración es un mapa provisto por un servicio de mapas en línea (Google Maps) y de la ubicación actual del usuario (GPS). Se integran diferentes proveedores de puntos de de interés (POIs). Los POIs se muestran en el mapa con diferentes símbolos. A medida que el usuario se mueve, el mapa se mantiene centrado en su posición y se muestra un rastro de su movimiento si el usuario así lo desea. Por el lado del cliente, la especificación Delivery Context Client Interfaces (DCCI) se utiliza para integrar la información de contexto de local obtenida a través de sensores con el navegador web móvil, que adapta el mashup a la localización actual del usuario.

En la Arquitectura presentada en la Figura 4 se puede observar la arquitectura del sistema TELAR, que de la misma manera que una aplicación mashup típica tiene tres niveles. El mashup se visualiza en la capa del cliente. El navegador carga la página web del mashup y ejecuta el código Javascript (AJAX) en el cliente. Esta página del mashup se carga desde el servidor de Mashup, que se encuentra en Internet en la capa servidor. Los datos ofrecidos por terceros proveedores que se utilizan se distribuyen por toda la web, estos se encuentran en la Capa de Proveedores de datos. Los mapas se cargan desde un servicio de mapas, que junto con los proveedores de datos, constituyen la capa de datos. Esta última capa se encuentra fuera de los límites de la organización del mashup.

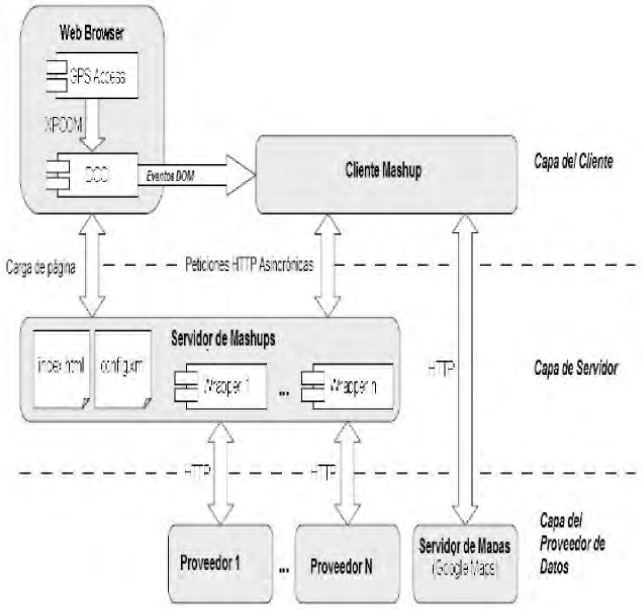


Figura 3. Componentes de TELAR

El cliente mashup utiliza lo desplegado en el servidor de mashup y estos pueden ser configurados/ parametrizados pero no necesitan ser programados por los usuarios finales. El cliente mashup lee asincrónicamente la configuración cuando la página del mashup se carga. Luego el cliente mashup construye la interfaz de usuario, esta muestra el mapa y visualiza los datos POIs de los proveedores de datos.

Una capa de normalización se utiliza lograr una interfaz consistente para la recuperación de datos utilizando wrappers, ya que este enfoque ofrece el más alto grado de flexibilidad. Para otros escenarios con menos diversidad, se pueden utilizar enfoques de normalización automática (por ejemplo, proveedores de datos que proporcionan canales RSS, como el usado en Yahoo Pipes [20]).

Tanto la información de contexto, como la ubicación del usuario, se integran al mashup, ampliando el navegador web. Para ello se utilizan dos componentes adicionales: el módulo DCCI y el módulo de acceso GPS. El módulo que implementa la especificación DCCI constituye la interfaz para proporcionar datos de contexto a las páginas web. El cliente mashup se registra como un detector de eventos en el módulo DCCI y este es notificado cada vez que se produce un cambio en la localización del usuario. El módulo de acceso GPS se conecta al dispositivo GPS y este pasa la información de ubicación necesaria al módulo DCCI.

Para implementar el cliente se utilizó el marco de trabajo Google Web Toolkit (GWT). Este dio la posibilidad de desarrollar un cliente mashup complejo en Java utilizando herramientas profesionales. Del lado del servidor, TELAR

requiere una aplicación servidor desde la cual el mashup se cargue y donde residan los wrappers de los proveedores de datos.

Resultados

Si bien existen herramientas para construir mashups sensibles al contexto en entornos de AmI estas son incipientes y todavía están en fase de prototipo. Aún no se cuenta con herramientas con la potencia y facilidad de uso de Yahoo Pipes para escenarios AmI.

Las tecnologías utilizadas, la forma de interacción con los usuarios, y de adquisición de conocimiento de este nuevo tipo de mashup, no están estandarizadas y existen múltiples estrategias y plataformas (como las mencionadas en la sección 4) para su construcción por lo que la mayoría de estas propuestas planteen una solución práctica para el escenario específico en cual son aplicadas.

Esto hace difícil integrar datos obtenidos desde sensores y utilizarlos para adaptar el contenido o el comportamiento de los mashups (y de cualquier aplicación web) ya que no existe un estándar de cómo capturar, representar y utilizar la información de contexto en las aplicaciones web.

Lo comentado anteriormente da lugar la necesidad de patrones, tanto para la adquisición de datos desde sensores (e información derivada) como en las estrategias para que estos datos sean aprovechados por los sistemas mashups de forma segura y que no comprometan la privacidad. Detallar sintéticamente los ejes del tema que se están investigando.

Como líneas futuras se proponen dos pilares, uno desde el punto de vista de las aplicaciones para la creación de mashups sensibles al contexto, se plantea la necesidad de contar con una herramienta del tipo de Pipes de Yahoo, pero que además de fuentes de datos RSS, APIS, y cajas de texto de entrada de datos (user input) pueda incorporar información de contexto provista por los sensores (context input) del dispositivo móvil del usuario de manera automática.

Siguiendo con esta línea como el segundo pilar sería interesante contar con un navegador web sensible al contexto, que de soporte a la AmI. Este Navegador de Web Ubicua (navegador sensible al contexto), se lo podría plantear como una evolución de un navegador tradicional como Mozilla Firefox o una extensión para los ya existentes. De esta forma se podría realizar una navegación de Ambientes Inteligentes. Esta extensión o mejora permitiría recabar datos de los sensores de dispositivos móviles de forma transparente para el usuario y las aplicaciones mashups e independientemente del hardware o tecnología subyacente, En este sentido se esta trabajando con HTML5 y formatos interoperables como ser REST y RSS y abstraer WSN (Redes de Sensores Inalámbricos por sus siglas en inglés) de la publicación de datos que ellas generan para lograr el objetivo.

Formación de Recursos Humanos

El equipo de trabajo se encuentra formado por un Doctor en Ciencias Informáticas, Un Doctorando en Ingeniería Telemática, un auxiliar de investigación graduado, y ocho auxiliares (Resolución rectoral 21/I/12) de investigación en período de realización de trabajos finales de grado. El número de tesinas de grado en curso con proyecto aprobado es tres y el número de trabajo de especialidad en curso con proyecto aprobado es uno. Los proyectos de grado se titulan "Plataforma para la publicación de datos de Redes de Sensores Inalámbricos, orientada a la visión de la Internet de las Cosas, Ambientes

Inteligentes y Mashups", "Diseño de un prototipo para monitoreo eficiente de iluminación basado en WSN utilizando HTML5" y "Contribución a la Gestión de Residuos Domiciliarios como una Aplicación en Ciudades Inteligentes". El Trabajo de Especialidad se titula "Plataformas para la creación de Mashups Sensibles al Contexto en Entornos de Inteligencia Ambiental"

Referencias

[1] Weiser M, The Computer for the 21st Century. Scientific American, 265(3):66–75, September 1991

[2] Weiser, M.. Some computer science issues in Ubiquitous Computing. Communications of ACM, 36(7), 75-84 1993

[3] Sosa, Eduardo O., Contribuciones al establecimiento de una red global de Sensores Inalámbricos. Tesis Doctoral. s.l.: Universidad Nacional de La Plata, Junio 17, 2011.

[4] Ahola J, Ambient Intelligence, ERCIM News, No 47, October 2001.

[5] Schilit, B., Theimer, M. Disseminating Active Map Information to Mobile Hosts. IEEE Network, 8(5) (1994) 22-32

[6] Pascoe, J. Adding Generic Contextual Capabilities to Wearable Computers. 2nd International Symposium on Wearable Computers (1998) 92-99

[7] Dey, A.K. Context-Aware Computing: The CyberDesk Project. AAAI 1998 Spring Symposium on Intelligent Environments, Technical Report SS-98-02 (1998) 51-54

[8] Gerd Kortuem, Zary Segall, and Martin Bauer. Context-aware, adaptive wearable computers as remote interfaces to 'intelligent' environments. In ISWC, pages 58–65, 1998.

[9] Salber, D., Dey, A.K., Abowd, G.D. Ubiquitous Computing: Defining an HCI Research Agenda for an Emerging Interaction Paradigm. Georgia Tech GVU Technical Report GIT-GVU-98-01 (1998)

[10] Anind Kumar Dey. Providing architectural support for building context-aware applications. PhD thesis, Georgia Institute of Technology, 2000. Director-Gregory D. Abowd.

[11] T. O'Reilly, "What Is Web 2.0, Design Patterns and Business Models for the Next Generation of Software", O'Reilly Media Inc., September 2005.

[12] A. Koschmider, V. Torres, V. Pelechano: Elucidating the Mashup Hype: Definition, Challenges, Methodical Guide and Tools for Mashups. 2nd Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009).

[13] Duane Merrill. Mashups: The new breed of Web app--Anintroduction to mashups. http://www.ibm.com/developerworks/xml/library/x-mashups.html

[14] Xuanzhe Liu; Yi Hui; Wei Sun; Haiqi Liang, "Towards Service Composition Based on Mashup," Services, 2007 IEEE Congress on , vol., no., pp.332-339, 9-13 July 2007

[15] Sitio de Internet: Screen Scrapper, http://www.screen-scraper.com/

[16] López de Ipiña D., Vázquez J.I. and Abaitua J., Context-Aware Mobile Mash-up for Ubiquitous Web, Puertollano, Spain, ISBN: 84-6901744-6, pp. 19-34, November 2006

[17] D. Lopez-De-Ipina, J. I. Vazquez, J. Abaitua. "A context-aware mobile mashup platform for ubiquitous web". Intelligent Environments, 2007. IE 07. 3rd IET International Conference on (2007), pp. 116-123.

[18] Andreas Brodt , Daniela Nicklas, The TELAR mobile mashup platform for Nokia internet tablets, Proceedings of the 11th international conference on Extending database technology: Advances in

database technology, March 25-29, 2008, Nantes, France

[19] A. Brodt, D. Nicklas, S. Sathish, and B. Mitschang. Contextaware mashups for mobile devices. In Web Engineering (WISE '08), pages 280–291. Springer-Verlag, 2008.

[20]Sitio de Internet: Yahoo Pipes, http://pipes.yahoo.com/pipes/