Universidad de las Ciencias Informáticas
FACULTAD 6

Título: Modelo de comunicación para las aplicaciones y componentes del Sistema de gestión y transmisión de contenidos audiovisuales.

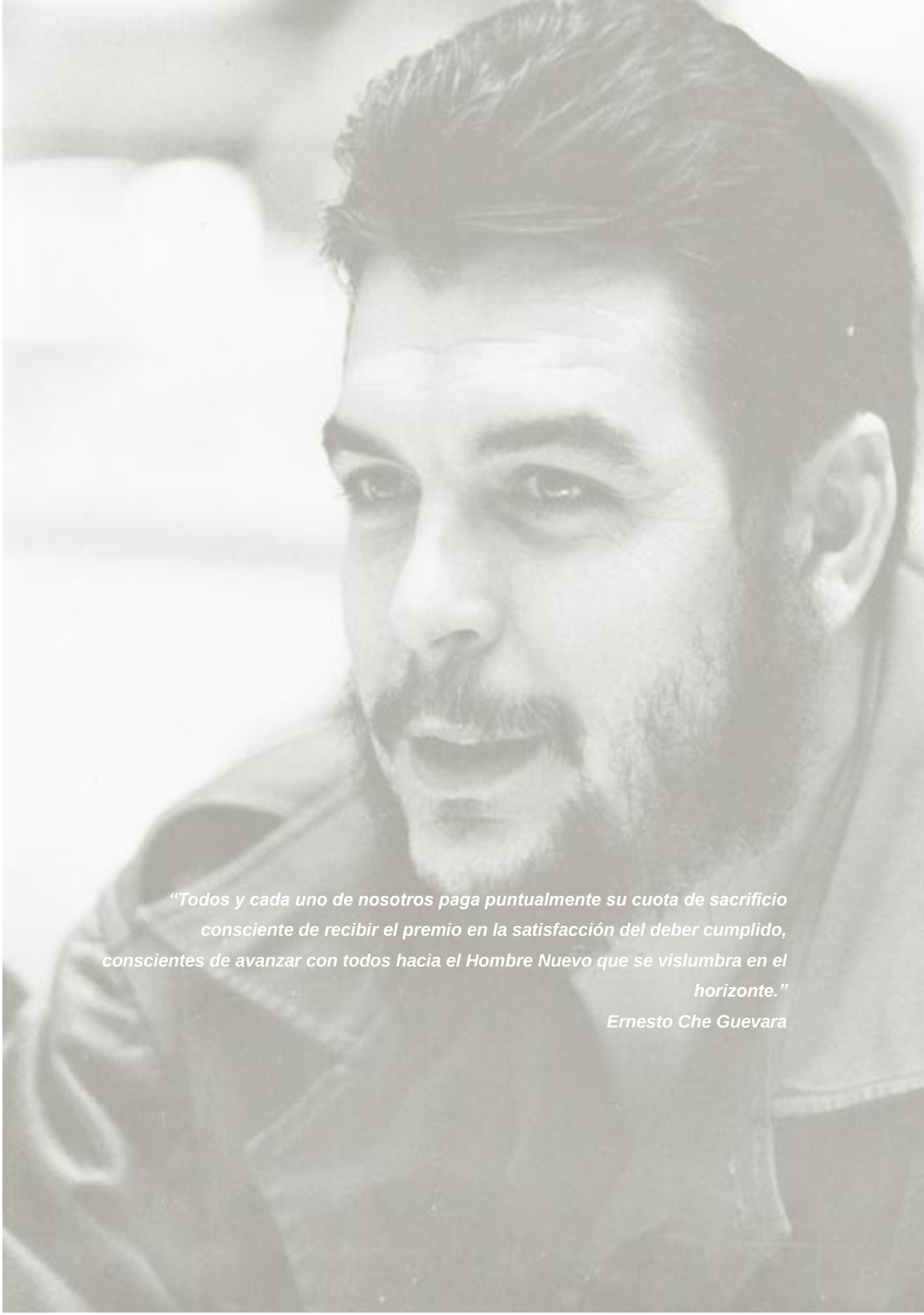
Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Ivón Martínez Genis

Tutor: Ing. Abel Díaz Berenguer

"La Habana, Junio de 2012"



“Todos y cada uno de nosotros paga puntualmente su cuota de sacrificio consciente de recibir el premio en la satisfacción del deber cumplido, conscientes de avanzar con todos hacia el Hombre Nuevo que se vislumbra en el horizonte.”

Ernesto Che Guevara

DECLARACIÓN DE AUTORÍA

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

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

Ivón Martínez Genis	**Abel Díaz Berenguer**
_____	_____
Firma del Autor	Firma del Tutor

DATOS DE CONTACTO

Tutor: Ing.Abel DíazBerenguer.

FormaciónAcadémica: IngenieroenCienciasInformáticas(Julio/2009).

CentroLaboral: UniversidaddelasCienciasInformáticas(UCI).

CorreoElectrónico:aberenguer@uci.cu

Dedicatoria

A mis padres, por ser ellos mi fuente de inspiración, guiarme por el buen camino y lo más importante porque los quiero con todo mi corazón.

A mi familia, porque sé que siempre me apoyarán y estarán para mí cuando lo necesite.

Estar hoy aquí se los debo a ustedes.

Agradecimientos

Mi mami, por toda la preocupación y esfuerzo en cuanto a mi bienestar.

Mi papito, que siempre se ha preocupado por mis estudios.

Mi hermanita, que me ha ayudado tanto, gracias, ahora me toca a mí.

A ellos por todo su amor y dedicación.

A toda mi familia, que todos han puesto su granito de arena, para formar lo que soy hoy día.

Mis amigas, por siempre estar para animarme cada vez que me derrumbo, todas las titas, Liset, Ilen y Anita.

A todas las personas que han compartido conmigo en la UCI, la gente del aula, las chicas del apartamento y a todos los amigos, incluyendo aquellos que ya no están en la universidad, pero que recuerdo con mucho cariño.

A los profesores que me han dedicado tiempo, imposible mencionarlos a todos, a Adnan, a los integrantes del tribunal y a mi tutor Abelito, que ha sido un amigo para mi durante el transcurso de la carrera.

A todos los que de alguna forma hicieron posible este trabajo y mi convivencia en la universidad.

Resumen

El avance en las tecnologías de redes, ha provocado que las aplicaciones necesiten desarrollarse, hasta lograr comunicarse entre sí mediante el intercambio de procesos. Actualmente existen diversas técnicas y procesos para alcanzar comunicación e interacción entre aplicaciones. Dentro de estas técnicas se observan el uso de estándares y servicios, de los cuales se hace un estudio en la presente investigación y se aplican los más factibles para obtener una buena interoperabilidad entre aplicaciones. Este trabajo propone un modelo, para estructurar como se logrará la comunicación entre los componentes del Sistema de gestión y transmisión de contenidos audiovisuales, SIAV, de modo que incremente la interoperabilidad entre los subsistemas del mismo. El modelo de comunicación propuesto, aprovecha las facilidades que ofrece la arquitectura basada en plug-ins, brindando un modelo de comunicación independiente de tecnologías, lo que propiciará mayor dinamismo en la solución. Por su lado las tecnologías de comunicación propuestas, presentan diversas características que las diferencian, pero cada una de ellas es factible de aplicar, ya que permiten comunicar las aplicaciones del sistema, haciéndolas más flexibles y dinámicas. Se construye un prototipo funcional que, respetando el modelo de comunicación propuesto, es capaz de lograr interoperabilidad entre los subsistemas existentes. Los resultados obtenidos mediante el método de expertos Delphi, corroboraron la efectividad de aplicar el modelo propuesto en SIAV.

Palabras claves: interoperabilidad, modelo, técnicas.

Índice de Contenido

Introducción

El mundo está siendo revolucionado por las nuevas creaciones tecnológicas y principalmente por la web y todo el ciberespacio. Este suceso se viene formando desde hace años y se encuentra cambiando la vida de las personas. Con un pequeño retroceso en el tiempo se puede ver como el ser humano desde sus inicios, ha sentido la necesidad de comunicarse, expresar sus pensamientos y emociones, e ir indagando en vías para facilitar su modo de vida. El correo, la radio, el cine, el teléfono, son medios de comunicación con objetivos de facilitar el intercambio de información.

Se puede hablar de redes y software informáticos, como avances tecnológicos de gran impacto en la sociedad. En la actualidad las aplicaciones juegan un papel significativo dentro de cada empresa informatizada. Estas últimas precisan de productos, que puedan estar compuestos por componentes de software, que presenten buena interoperabilidad. En busca de un mejor rendimiento y eficacia, las aplicaciones informáticas, comienzan a convertirse en aplicaciones distribuidas, ya que operan con datos ubicados en diferentes nodos de procesamiento. Esto precisa que surja la necesidad de contar con una adecuada interoperabilidad. Las aplicaciones distribuidas, son capaces de llevar a cabo prestaciones de una computadora a otra, de compartir recursos y dar solución a distintas necesidades.

Los desarrolladores de aplicaciones distribuidas, pueden encontrase con numerosos obstáculos relacionados con cambios constantes de los requisitos del negocio, versiones posteriores al producto final, especificaciones no explícitas, variedad de plataformas a conectar, distintas formas de interacción y demás problemas de compatibilidad. Atendiendo a estos inconvenientes, muchos investigadores se han dedicado a elaborar técnicas para facilitar la interoperabilidad entre componentes. Comenzando por mecanismos de comunicación basados en socket, los cuales brindan una interfaz que permite a los programadores hacer uso de la red para comunicar aplicaciones. Se pueden mencionar además, el trabajo con estándares y la utilización de servicios como procesos necesarios para comunicar aplicaciones.

Los estándares han permitido la interacción entre aplicaciones informáticas, definiendo como interactuarán los diferentes componentes informáticos. Los estándares permiten interoperabilidad entre diferentes aplicaciones o servicios (1).

Los servicios consisten en la capacidad de desarrollar una interfaz que permita la relación entre dos sistemas informáticos, utilizando una colección de estándares y protocolos. Permiten llevar a cabo un intercambio de información y transacciones entre dichos sistemas, sin importar lenguaje de programación, ni plataforma en que se encuentran desarrollados (2). Puesto que los servicios están diseñados para ser independientes, autónomos y para interconectarse adecuadamente, pueden combinarse en aplicaciones complejas respondiendo a las necesidades de cada momento (3).

La utilización de estándares y servicios web brinda óptimos resultados. En lugar de tratar con múltiples aplicaciones incompatibles en varios equipos y lenguajes de programación, es posible agregar una capa de abstracción, basada en estándares y fácil de integrar con prácticamente cualquier entorno. De esta manera se obtienen aplicaciones más flexibles y se refuerza la interoperabilidad entre estas.

Cuba trabaja arduamente para llevar conocimientos de informática, a distintas esferas de la sociedad. La formación de las personas, incursionando desde la enseñanza primaria, hasta la universitaria, ha permitido un avance en la industria de la informática. Fuera de las escuelas también se han preparado a todos los interesados, mediante cursos de computación, software educativo y la creación de los Joven Club de Computación y Electrónica. En Cuba existen además, universidades comprometidas en la formación de nuevos ingenieros informáticos, una de estas escuelas encargadas de forjar profesionales es la Universidad de las Ciencias Informáticas (UCI).

En Cuba la formación de las personas, ha sido una de las cosas que se han puesto en práctica para lograr un avance en la industria de la informática. Atendiendo a esto se ha inculcado interés en la sociedad por esta ciencia, incursionando desde la enseñanza primaria hasta la universitaria. Uno de los pilares que se puede encontrar dentro de este proceso de informatización es la universidad de ciencias informáticas, creada con el objetivo de formar nuevos ingenieros informáticos.

La UCI posee un modelo de formación de profesionales encaminado al proceso de investigación, innovación y desarrollo, además de sus procesos docentes característicos. Esta universidad cuenta con varias facultades, dirigidas a la investigación y producción de software. En la facultad seis, se encuentra el Centro de Geoinformática y Señales Digitales (GEySED) que se dedica a brindar soluciones en

estas áreas temáticas. Específicamente en el departamento de Señales Digitales, se ha desarrollado el Sistema de gestión y transmisión de contenidos audiovisuales (SIAV). Este sistema se encarga de automatizar procesos relacionados con la gestión, procesamiento y transmisión de contenido multimedia.

SIAV surge luego de la unión de varios productos previamente desarrollados en el departamento, los cuales estaban realizados en diferentes plataformas y lenguajes de programación. Fue necesario realizar la integración de los productos mencionada anteriormente, a través de la unión de sus modelos de datos. De esta manera se logró interoperabilidad en SIAV, pero no presenta buena escalabilidad, existe dependencia de la base de datos, implicó tiempo de desarrollo innecesario y reelaboración del código existente, disminuyendo el rendimiento en cuanto a la interacción entre aplicaciones de SIA.

La presente investigación, surge por la necesidad que existe de darle solución a la situación problemática antes expuesta; por lo que el problema de investigación que se plantea es: ¿Cómo propiciar la interacción entre los subsistemas del Sistema de gestión y transmisión de contenido audiovisuales y con aplicaciones externas? Tomando como base el problema definido anteriormente, se plantea como objeto de estudio: Técnicas y procesos de interoperabilidad entre componentes y aplicaciones. Se establece como campo de acción: La interoperabilidad entre los subsistemas de SIAV y a su vez con aplicaciones externas. Se tiene como objetivo general: Desarrollar el modelo de interoperabilidad para el Sistema de gestión y transmisión de contenido audiovisuales. Se define como idea a defender: Si se desarrolla un modelo de interoperabilidad para el Sistema de gestión y transmisión de contenidos audiovisuales se propiciará la interacción entre los subsistemas de SIAV y a su vez con aplicaciones externas.

Para dar cumplimiento al objetivo planteado en el presente trabajo se definieron las siguientes Tareas de investigación:

1. Caracterizar los procedimientos y técnicas de interoperabilidad entre componentes y aplicaciones.
2. Caracterizar soluciones o aplicaciones que garanticen interoperabilidad entre sus componentes de software.
3. Describir la propuesta de solución.
4. Determinar las herramientas y tecnologías de desarrollo a utilizar, para

implementar un prototipo funcional basado en la solución propuesta.

5. Validar la propuesta de solución mediante el desarrollo de un prototipo funcional.

Diseño metodológico

Métodos teóricos:

Analítico-Sintético: para recopilar toda la información posible referente a la interoperabilidad entre aplicaciones, para valorar y caracterizar los procedimientos y técnicas de interoperabilidad.

Histórico-Lógico: Para determinar la evolución histórico-lógica de las técnicas y procedimientos de interoperabilidad entre aplicaciones informáticas.

Métodos Empíricos:

Entrevista: Realizar entrevistas para obtener información real sobre las necesidades existentes en la plataforma. Para esto se toma como población a los profesores de SIAV, resultando una población de 10 miembros. Se seleccionó como muestra, 5 personas debido a su experiencia, que representa el 50 % de la población. Para esto se utilizó la técnica de muestreo no probabilístico, específicamente del tipo intencional, tomándose como criterio de selección de la muestra, que los profesores tuvieran conocimientos de interoperabilidad entre aplicaciones.

El presente trabajo se encuentra estructurado de la siguiente manera:

En el Capítulo 1 se exponen los conceptos elementales relacionados con la investigación, para facilitar el entendimiento de la misma e igualmente revelar su importancia. Se detalla la situación problemática actual. Se caracterizan tecnologías asociadas al objeto de estudio de investigación. Conjuntamente a esto, se mencionan soluciones existentes para garantizar interoperabilidad entre aplicaciones y componentes de software.

En el Capítulo 2 se describe la propuesta de solución, detallando como estará estructurado el modelo y determinando pasos y procedimientos lógicos que se deben seguir para su utilización.

En el Capítulo 3 se realiza la validación del modelo propuesto. Se realizan encuestas para validar la propuesta de solución, utilizando el método Delphi y se exponen los

resultados del mismo.

CAPÍTULO 1: Fundamentación Teórica

El siguiente capítulo tiene como objetivo caracterizar los procedimientos y técnicas de interoperabilidad entre aplicaciones. Se definen además los conceptos asociados al objeto de estudio: Técnicas y procesos de interoperabilidad entre componentes y aplicaciones. Se argumenta sobre la situación problemática, para entender las causas que originan la presente investigación. Se caracterizan soluciones que garantizan interoperabilidad entre sus componentes de software.

1.1. Conceptos Asociados

1.1.1. Aplicaciones

Una aplicación informática es un tipo de software que permite al usuario realizar uno o más tipos de trabajos (4). En la informática, se entiende como aplicación, un programa informático, donde se permite la interacción entre el usuario y la computadora. En una aplicación, el usuario tiene la opción de elegir opciones y ejecutar acciones.

1.1.2. Componente

Un componente es una unidad autónoma de composición de aplicaciones de software. Poseen un conjunto de interfaces y un conjunto de requisitos. Puede ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio (5). Se concluye que un componente es una pequeña parte, independiente y reemplazable de una aplicación.

1.1.3. Interoperabilidad

Capacidad de un sistema de software de comunicarse y compartir datos, información, documentos y objetos digitales de forma efectiva (con una mínima o nula perdida de su valor y funcionalidad), con uno o varios sistemas de información (siendo generalmente estos sistemas completamente heterogéneos, distribuidos y geográficamente distantes), mediante una interconexión libre, automática y transparente, sin dejar de utilizar en ningún momento la interfaz del sistema propio (6).

Componentes de la interoperabilidad (7):

Interoperabilidad técnica: Intercambiar información. Recoge cuestiones de conectividad y uso de los sistemas y de aplicaciones de servicios.

Interoperabilidad semántica: Poder entenderse. Se dedica a conseguir que los

sistemas estén preparados para leer y entender la información intercambiada.

Interoperabilidad organizativa: Que sea útil y se utilice. Concierne al establecimiento de estrategias y a su despliegue para que diferentes organismos puedan conectarse entre sí e intercambiar información y trámites.

Se define brevemente interoperabilidad como la capacidad que pueden presentar las aplicaciones, para intercambiar información.

1.1.4. Sistemas

Un sistema se define como un conjunto de entidades caracterizadas por ciertos atributos, que tienen relaciones entre sí y están localizadas en un cierto ambiente, de acuerdo con un cierto objetivo (8). Finalmente se concluye que no es más que una serie de entes funcionando como un todo.

1.1.5. Sistemas distribuidos

Conjunto de procesos que se ejecutan concurrentemente en una o más computadoras y se comunican intercambiando mensajes. Estos se perciben como un sistema único. Pueden compartir recursos, presentan capacidad de crecimiento y tolerancia a fallos (9).

Se concluye que un sistema distribuido no es más que un conjunto de aplicaciones, que aparentan ante el usuario como una sola aplicación. Esto es posible gracias a la comunicación que establecen mediante la red, permitiendo que trabajen como un solo sistema.

1.1.6. Técnicas

Las técnicas son utilizadas para describir acciones regidas por normas que tienen el propósito de arribar a un resultado específico, tanto a nivel científico como tecnológico, artístico o de cualquier otro campo (10). Son procedimientos que tienen como objetivo alcanzar un resultado determinado.

1.1.7. Procesos

Instancias de ejecución de un programa. Conjunto de actividades mutuamente relacionadas o que interactúan, las cuales transforman elementos de entrada en resultados (11).

1.1.8. Comunicación

La comunicación es el acto por el cual una entidad establece con otra un contacto que le permite transmitir una información (12). Es un proceso que permite la interoperabilidad entre aplicaciones.

1.1.9. Modelo

Un modelo es una representación de un objeto, sistema o idea, de forma diferente al de la entidad misma. El propósito de los modelos es ayudar a explicar, entender o mejorar un sistema. Un modelo de un objeto puede ser una réplica exacta de éste o una abstracción de las propiedades dominantes del objeto (13).

Se puede concluir que un modelo no es más que una manera sencilla de representar algo. Un modelo puede ser una imagen, un conjunto de pasos a seguir, un gráfico, un esquema o una explicación sencilla que ayude a entender lo que se debe hacer.

1.2. Objeto de estudio: Técnicas y procesos de interoperabilidad entre componentes y aplicaciones

1.2.1. Descripción general

Para un usuario es imperceptible la diferencia entre utilizar técnicas de interoperabilidad o componentes tradicionales. En la actualidad existen diversas tecnologías desarrolladas específicamente para comunicar aplicaciones. Estas logran interoperabilidad entre aplicaciones a través de la publicación de servicios, utilización de middleware y mediante el uso de estándares y protocolos de comunicación.

1.2.1.1. Servicios

Un servicio es una función, que puede ser descubierta y descrita en la red. Los servicios realizan tareas concretas, pueden corresponder a un proceso de negocio tan sencillo como introducir o extraer un dato. Los servicios pueden acoplarse dentro de una aplicación para permitir que proporcione funcionalidades (3).

Según la W3C1, un servicio es un recurso abstracto que representa una capacidad de

[1]World Wide WideConsortium, Es un consorcio reconocido a nivel internacional donde las organizaciones miembros trabajan conjuntamente para desarrollar estándares. Su misión es guiar la Web hacia su máximo potencial a través del desarrollo de protocolos y pautas.

realizar tareas, que conforman una funcionalidad coherente desde el punto de vista de las entidades proveedoras y consumidoras. Para ser usado, un servicio debe ser realizado por un agente proveedor concreto. Se definen además como aplicaciones auto-contenidas que pueden ser descritas, publicadas, localizadas e invocadas sobre cualquier red (14).

Los servicios son independientes, encapsulan sus propios datos y no forman parte de la aplicación, sino que son utilizados por ella. Los servicios pueden ser generados por distintos equipos de desarrollo, en diferentes plataformas y en diferentes espacios y tiempos. Una aplicación puede ir creciendo y aumentando su funcionalidad a medida que se construyan nuevos servicios. Estos servicios, no necesariamente deben estar bajo el control del equipo de desarrollo de la aplicación. Es esencial que entre las aplicaciones y los servicios que ellas consumen, exista un vínculo satisfactorio (15).

Se puede concluir que un servicio proporcionado por un servidor no es más que un conjunto de operaciones útiles para los usuarios. Dentro de estos se encuentran los servicios web, sistemas de software diseñados para mantener interacciones entre aplicaciones sobre una red. Tiene una interfaz descripta en un formato procesable por una máquina. Puesto que los servicios están diseñados para ser independientes, autónomos y para interconectarse adecuadamente, pueden combinarse y recombinarse con suma facilidad en aplicaciones complejas que respondan a las necesidades de un proyecto (14).

Los servicios web son aplicaciones que utilizan estándares para el transporte, codificación y protocolo de intercambio de información. Los servicios web permiten la intercomunicación entre sistemas de cualquier plataforma y se utilizan en una gran variedad de escenarios de integración (3).

Se observa que un servicio web es una aplicación de software que intercambia datos con otras aplicaciones en otras computadoras. Permite reutilización de métodos y funciones, de una aplicación a otra.

Los sistemas distribuidos modernos tienen una capa adicional de código entre la capa de presentación y la capa de negocio: la de servicios. Esta se utiliza para exponer las distintas funciones con las que se pueden interactuar. La capa de servicios en una aplicación distribuida, representa una capa adicional de código que crea un límite entre otras capas. Es una forma de software implementada a modo de interfaz que mantiene

la implementación transparente para los consumidores (16).

Internamente, cada servicio se compone de componentes de software, al igual que cualquier otra aplicación y estos componentes pueden ser agrupados de forma lógica en la capa de presentación, de negocios y de datos. Otras aplicaciones pueden hacer uso de los servicios sin ser conscientes de la forma en que se implementan. Cuando una aplicación debe prestar servicios a otras aplicaciones, lo mejor sería utilizar una capa de servicios, que expone la funcionalidad de negocio de la aplicación, como se muestra en la Figura 3. La capa de servicios ofrece de manera eficaz, una visión alternativa, que permite a los clientes utilizar un canal diferente para acceder a la aplicación(17).

En este escenario, los usuarios pueden acceder a la aplicación a través de la capa de presentación, que se comunica directamente con los componentes en la capa de negocio, o mediante una aplicación en la fachada de la capa de negocio, si los métodos de comunicación requieren composición de la funcionalidad. Mientras tanto, los clientes externos y otros sistemas pueden acceder a la aplicación y hacer uso de su funcionalidad mediante la comunicación con la capa de negocio a través de interfaces de servicio. Esto permite a la aplicación mejorar el soporte de múltiples tipos de clientes y promueve la reutilización y el alto nivel de composición funcionalidad en las aplicaciones(17).

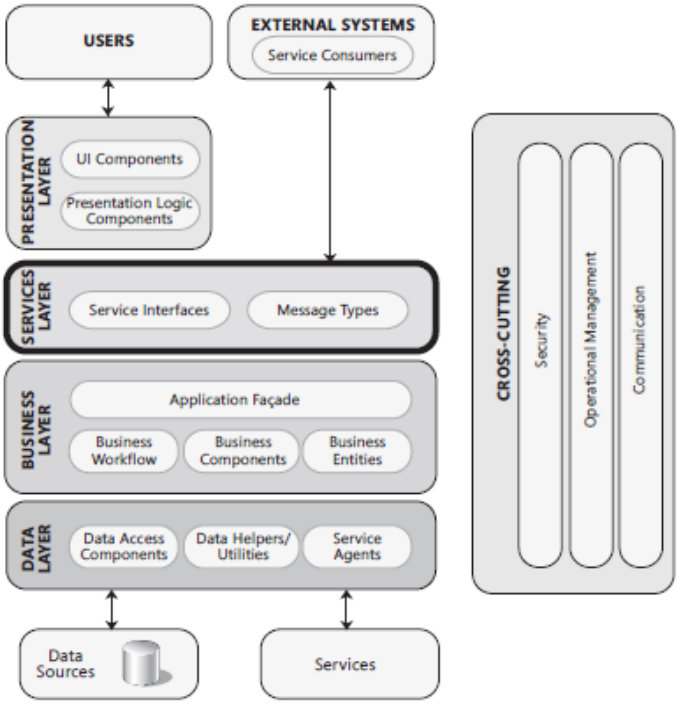


Figura 3. La incorporación de una capa de servicios en una aplicación(17).

Las capas representan la separación lógica de los componentes de una aplicación en grupos que representan distintos roles y funciones. Se utilizan capas para mejorar el mantenimiento de la aplicación y hacer que sea más fácil de escalar cuando sea necesario, mejorando así el rendimiento.

1.2.1.2. Protocolos y estándares de comunicación

Los servicios pueden ser brindados mediante el uso de protocolos y estándares de comunicación. Los protocolos de comunicación establecen un mecanismo por el cual, un proveedor y un consumidor se pueden comunicar. Un protocolo es un conjunto de reglas de comunicaciones, que se implantan entre las aplicaciones a comunicar. Los protocolos gobiernan el formato, sincronización, secuencia y control de errores. A continuación se presentan funciones básicas de un protocolo (18):

Control de llamada: Establecimiento de conexión entre fuente y destino, esta función lleva a cabo el mantenimiento y monitoreo de la conexión y los procedimientos de conexión y desconexión de una llamada, transferencia de datos, videoconferencia, entre otras.

Control de error: Verificación y control de errores durante la transmisión a través de algoritmos de verificación y control de errores.

Control de flujo:

- Manejo de contención de bloques.
- Regulación del tráfico.
- Retransmisión de bloques.
- Convenciones para direccionamiento.
- Control por pasos y de extremo a extremo.

1.2.1.2.1. Llamada a procedimiento remoto (RPC)

Es una infraestructura cliente / servidor que permite a una aplicación distribuida correr en múltiples máquinas heterogéneas (19).En este caso, la distribución consiste en mover rutinas del cliente y colocarlas en el programa servidor. En el cliente, las rutinas son reemplazadas por otras con igual nombre y parámetros, pero con código generado por RPC llamado stub2. Este código se encarga de efectuar la llamada remota o sea, contactar al programa servidor, solicitar la ejecución del procedimiento y esperar la respuesta del mismo (20).

Entre sus ventajas, se cuentan la interoperabilidad, portabilidad y flexibilidad. Sin embargo, no es el middleware indicado para aplicaciones que involucren objetos distribuidos o para desarrollar con lenguajes orientados a objetos ya que presenta todas las características de los lenguajes procedurales.

1.2.1.2.2. Protocolo simple de acceso a objeto (SOAP)

Protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML (21). El lenguaje de marcado extensible (XML), es usado para estructurar información en cualquier fichero que contenga texto. Es una estructura jerárquica, exigente en la sintaxis y en la organización del documento, lo que hace más fácil la construcción de bibliotecas (22).

SOAP permite el desarrollo de aplicaciones distribuidas, soporta diferentes aplicaciones y permite llamadas a procedimientos remotos (RPC) a través de HTTP. No se preocupa por el sistema operativo, ni lenguaje de programación (23).

[2]Parte de código usado como sustituto de alguna otra funcionalidad, puede simular comportamiento de código existente o ser el sustituto temporal para un código aun no desarrollado.

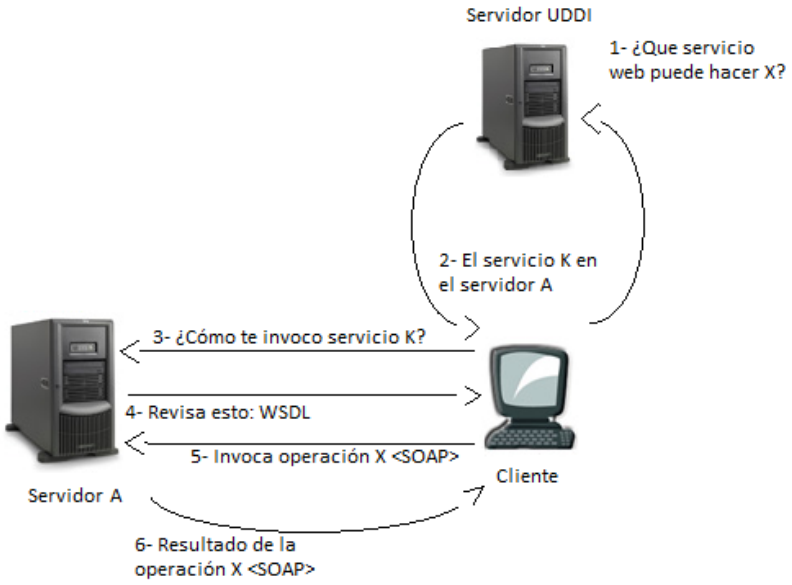


Figura 1. Arquitectura cliente/servidor en SOAP.

SOAP necesita especificar cómo se acceden a los servicios, las funciones que dispone y los argumentos de entrada y salida. Como se observa en la figura 1, utiliza un lenguaje neutro para describir las funciones residentes en el servidor. Este es el lenguaje de descripción de servicios web (WSDL), facilita a los clientes entender las funcionalidades del servidor, conocer como invocar los servicios y describirlos utilizando un documento XML. Para brindar los servicios se necesita un registro donde almacenarlos, para esto se utiliza un directorio de software, UDDI (Descripción universal de descubrimiento e integración) donde se publican e investigan los servicios, así como la información técnica de sus características y las reglas para el uso de los mismos.

Algunas de las Ventajas de SOAP son (23):

- No está asociado con ningún lenguaje: Los desarrolladores pueden elegir el lenguaje de programación.
- No se encuentra fuertemente asociado a ningún protocolo de transporte: La especificación de SOAP no describe como se deberían asociar los mensajes de SOAP con HTTP. Un mensaje de SOAP no es más que un documento XML, por lo que puede transportarse utilizando cualquier protocolo capaz de transmitir texto.
- No está atado a ninguna infraestructura de objeto distribuido.

- Aprovecha los estándares existentes en la industria: Utiliza XML y HTTP.
- Permite la interoperabilidad entre múltiples entornos.

1.2.1.2.3. XML-RPC

Proporciona un mecanismo sencillo de XML y HTTP que permite hacer llamadas a funciones a través de la red. XML-RPC ofrece simples conjuntos de herramientas para la conexión de sistemas dispares. Utiliza el protocolo HTTP para transmitir información desde un equipo cliente a un equipo servidor, describe la naturaleza de las solicitudes y de las respuestas con un vocabulario XML. Los clientes especifican un nombre de procedimiento y los parámetros en la solicitud de XML y el servidor devuelve un error o una respuesta XML (24).

En XML-RPC el cliente ejecuta un RPC en el servidor y el servidor procesa cada RPC llamando el correspondiente manejador de RPC. Siempre se habla en términos de cliente/servidor, existe un sistema que realiza la solicitud (el cliente) y otro que la atiende (el servidor) y como es de imaginarse un elemento clave en ambos puntos es: el parser XML, el cual se encarga de codificar la información al lenguaje XML para que pueda ser enviada entre el cliente y el servidor (25).

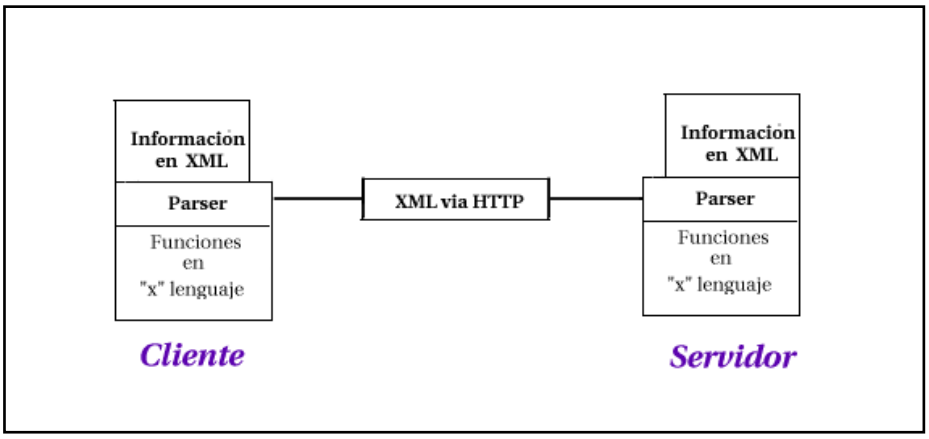


Figura 2.Arquitectura cliente/servidordeXML-RPC (25).

Concretamente, XML-RPC utiliza peticiones POST de HTTP para enviar los mensajes, en formato XML, indicando (26):

- El procedimiento que se va a ejecutar en el servidor.
- Los parámetros que se necesitan para ejecutar dicho procedimiento.
- El servidor devolverá el resultado en formato XML.

XML-RPC maneja un conjunto de tipos para el paso de parámetros, valores de retorno y errores. Estos tipos se representan mediante etiquetas XML. Existen tipos básicos y tipos compuestos, como se observan en la tabla 1 (26):

Tipos básicos:	
<int> o <i4>	Enteros de 32 bits.
<double>	Coma flotante de 64 bits.
<boolean>	Verdadero o falso.
<string>	Texto ASCII (y UNICODE).
<dateTime.iso8601>	Fecha en formato ISO8601.
<base64>	Binario codificado en base 64.
Tipos compuestos:	
<array>	Información secuencial.
<struct>	Contenido desordenado, identificable por nombre de campo.

Tabla 1. Tipos de datos (26).

Estructura de XML-RPC (27):

XML-RPC consiste de tres pequeñas partes, que combinadas definen una llamada completa a un procedimiento remoto:

Modelo de datos XML-RPC: Es el conjunto de tipos de datos usados para pasar parámetros, devolver resultados y generar códigos de error.

Estructura XML-RPC llamada: Es un HTTP POST request que contiene información del método y los parámetros.

Estructura XML-RPC respuesta: Es un HTTP response que contiene el valor resultante o información de error, según el caso.

XML-RPC llamada:

- Los XML-RPC requests son una combinación de headers HTTP con contenido XML.

- Cada llamada contiene un único documento XML cuyo elemento raíz es un methodCall, que a su vez contiene un elemento methodName que identifica el procedimiento a llamar y un elemento params que especifica los parámetros y sus valores.

XML-RPC respuesta:

- Los XML-RPC responses son muy parecidos a las llamadas pero con un par de detalles extra: El elemento methodCall se reemplaza por el elemento methodResponse y desaparece el elemento methodName.
- Si el llamado fue exitoso (es decir: El procedimiento fue encontrado, se ejecutó correctamente y devolvió un resultado), el elemento methodResponse contiene el resultado del procedimiento.
- Si en cambio hubo algún problema, el elemento methodResponse contiene un elemento fault (en lugar de params) que indica el error encontrado.

Al estar basado en XML, XML-RPC es independiente de plataforma y de lenguaje de programación (26).Aunque es una de las primeras tecnologías creadas para brindar servicios, no deja de ser útil, siendo su simplicidad es su principal ventaja. En SIAV, XML-RPC aporta grandes beneficios, permite la comunicación entre cliente y servidor, logrando sistemas distribuidos interoperables y flexibles.

1.2.1.3. Arquitectura orientada a servicios (SOA)

Siguiendo la arquitectura orientada a objetos, se puede lograr buena interoperabilidad entre las aplicaciones, ya que trabaja mediante el uso de servicios. La arquitectura SOA establece un marco de diseño para la integración de aplicaciones independientes de manera que desde la red pueda accederse a sus funcionalidades, las cuales se ofrecen como servicios. La forma más habitual de implementarla es mediante servicios web (3).

Con el fin de despuntar las utilizaciones de las aplicaciones y con unas mejoras tributadas a la programación orientada a objetos, se comenzó a desarrollar el concepto de arquitectura orientadas a servicios. SOA ofrece la posibilidad de llamar a un componente de negocios desarrollado en una plataforma X desde una aplicación corriendo en cualquier plataforma, en cualquier parte del mundo, utilizando para ello protocolos y estándares como SOAP, XML y HTTP.

Se define SOA como: “Conjunto de componentes que pueden ser invocados, cuyas descripciones de interfaces se pueden publicar y descubrir” (14).

Todas las arquitecturas SOA tienen las siguientes características claves (28):

- Los servicios SOA auto describen su interfaz en documentos XML que son independientes de la plataforma. WSDL es el lenguaje utilizado para describir los servicios. Además describe laespecificación de la finalidad, funcionalidad, forma de uso y restricciones del servicio.
- Los servicios SOA se comunican conmensajes formalmente definidos vía un esquema XML. La comunicación entre consumidores y proveedores de servicios típicamente ocurre en un ambiente heterogéneo, con poco o ningún conocimiento sobre el proveedor. Se puede utilizar XML-RPC como una capa de transporte para enviar mensajes entre los consumidores de servicios y proveedores.
- Los servicios SOA contiene un registro UDDI para registrar y buscar servicios.Proporciona además, facilidades para descubrir servicios y adquirir la información necesaria para su uso: Información de contrato, localización, personas de contacto, acuerdos a nivel de servicio, descubrimiento dinámico.

UnproyectoSOAbienejecutadopermitealinearlosrecursosdetecnologías informáticasdeformamásdirectaconlosobjetivosdenegocio.
Aportaunmayorgradodeintegraciónconclientesyproveedores.
Proporcionaunainteligenciadenegociomásprecisaymásaccesibleconlacualsepodránadoptarmejoresdecisiones.
Ayudaalasempresasaoptimizarsusprocesosinternosysusflujosdeinformaciónparamejorarlaproductividadindividual.Elresultadonetoesunaumentomuynotabledelaagilidaddelaorganización (3).

Adoptando una Arquitectura Orientada a Servicios se obtendrá un menor coste de mantenimiento, una mayor eficiencia en la adaptación, capacidad de respuesta y despliegue que respondan a las demandas de los usuarios. Pero resulta un proceso muy complejo para adoptar en un proyecto, ya que cada aplicación existente, tendría que aplicar una arquitectura SOA. SOA es una arquitectura que bien estudiada puede aportar grandes beneficios, aunque esto no implica que pueda ser utilizada en cualquier escenario. Por tanto a opinión de la autora del presente trabajo, no sería factible aplicar una arquitectura SOA.

1.2.1.4. Middleware

Los middleware han hecho disponible la programación distribuida a los desarrolladores de aplicaciones. Realizan aplicaciones distribuidas sin necesidad de tener que ser un experto en redes, ya que el middleware se encarga de los principales puntos de comunicación, tales como codificación y decodificación de los datos para la transmisión (29).

El middleware se define como un software distribuido constituido por un conjunto de recursos y servicios que permiten a múltiples procesos, corriendo en una o más máquinas, interactuar a través de una red. Es una colección no estructurada de recursos y servicios, ubicados entre la capa de transporte y la capa de aplicación, que pueden ser utilizados individualmente o conjuntamente por los procesos de la aplicación (20).

Se puede entender un middleware como, software de conectividad que hace posible que aplicaciones distribuidas puedan ejecutarse sobre distintas plataformas heterogéneas, es decir, sobre plataformas con distintos sistemas operativos, que usan distintos protocolos de red y que incluso, involucran distintos lenguajes de programación en la aplicación distribuida. Desde otro punto de vista, un middleware se puede entender como una abstracción en la complejidad y en la heterogeneidad que las redes de comunicaciones imponen. De hecho, uno de los objetivos de un middleware es ofrecer un acuerdo en las interfaces y en los mecanismos de interoperabilidad, como contrapartida de los distintos desacuerdos en hardware, sistemas operativos, protocolos de red y lenguajes de programación (30).

De acuerdo a la funcionalidad ofrecida y al criterio de distribución de recursos y servicios, se pueden definir distintos tipos de middleware:

1.2.1.4.1. Extensión de lenguajes comunes (CLE)

Software de conectividad distribuida, el cual define los objetos administrados y distribuye sus definiciones a un sitio remoto para mejorar la flexibilidad y la eficiencia. CLE es una plataforma de extensión común. Los módulos desarrollados en CLE pueden recibir llamadas desde cualquiera de los otros lenguajes compatibles. Además, CLE también presenta un patrón general para llamadas mixtas. CLE es multi-plataforma, es compatible con Win32 y Linux X86 (31).

CLE es compatible con la técnica de objetos distribuidos, la cual utiliza los objetos

como medios para implementar llamadas mixtas entre lenguajes. El objeto se presenta como una memoria estructurada y una lista de punteros a función. CLE es solamente una biblioteca compartida, la cual no impone ningún tipo de restricción referente al uso de un lenguaje de secuencias de comandos especial y puede emplearse para desarrollar tipos de aplicaciones distribuidas (31).

A continuación se mencionan las principales prestaciones de CLE (31):

- Permite las llamadas mixtas entre diferentes lenguajes, tales como C++, Lua, Python, Java, C#, java y PHP y puede extenderse a otros lenguajes con facilidad.
- Permite el desarrollo de extensiones o módulos de bibliotecas con C++, Lua, Python, c#, Java, PHP, las cuales se pueden utilizar en cualquiera de los lenguajes compatibles.
- Presenta herramientas completas, que incluyen carga de servicios, creación de paquetes, publicación, depuración, administración y configuración, compatibles con las versiones de Win32 y Linux.
- Simplifica el desarrollo de servicios web, genera WSDL (Lenguaje de Descripción de Servicios web) de forma automática, permite el desarrollo de servicios web con C, C++, Lua, Python, PHP, Java, C#.
- Completa función de administración de objetos, los objetos no sólo tienen atributos, funciones y eventos sino que también pueden contener secuencias de comandos y datos.
- Tecnología de llamada remota basada en objetos, compatibles con llamadas sincrónicas y llamadas de clientes asíncronas.
- Presenta los códigos fuente de los módulos a Python, Java, PHP y C#.

Esunsoftwarepropietarioycostoso,porlocualnoesviablesueleciónparatrabajarenlasoluci óndelapresenteinvestigación.

1.2.1.4.2. Arquitectura intermediaria para solicitar objetos comunes (CORBA)

Es un middleware que permite a los clientes invocar operaciones de objetos distribuidos, sin preocuparse por la localización de objetos, lenguaje de programación, plataforma de sistema operativo, hardware ni por los protocolos de comunicación y las interconexiones. Incluye servicios para seguridad, transacciones, notificaciones y otros

(32).

Está estandarizado por el OMG3, presenta una buena infraestructura para construir aplicaciones distribuidas e integrar aplicaciones heterogéneas (33).

CORBA es una tecnología que oculta la programación a bajo nivel de aplicaciones distribuidas, de tal forma que el programador no se tiene que ocupar de tratar con sockets o flujos de datos. No obstante CORBA también brinda al programador una tecnología orientada objetos, las funciones y los datos se agrupan en objetos. Estos objetos pueden estar en diferentes máquinas, pero el programador accederá a ellos a través de funciones normales dentro de su programa. Los métodos y datos CORBA se agrupan formando lo que se denominan interfaces, las interfaces pueden ser interpretadas como objetos que agrupan datos y métodos para acceder a estos. Todas estas interfaces se definen usado un lenguaje IDL para la definición de interfaces (Interface DefinitionLanguage). Este lenguaje es estándar y lo soportan todas las implementaciones CORBA (34).

Aunque es un buen camino para lograr integrar aplicaciones, no es una tecnología sencilla de utilizar y sus implementaciones, tardan en concluirse.

1.2.1.4.3. Motor de comunicación de internet (ICE)

ICE es una plataforma middleware orientada a objetos, que surge de CORBA, contiene las mismas funcionalidades de este y muchas más, además de ser mucho más fácil de utilizar y aprender. Provee herramientas, APIs y soporte de bibliotecas para construir aplicaciones cliente servidor orientadas a objetos. Las aplicaciones ICE son ideales para usar en entornos heterogéneos. El cliente y el servidor pueden ser escritos en diferentes lenguajes de programación, pueden ejecutarse en diferentes sistemas operativos y en máquinas con arquitecturas diferentes y pueden comunicarse usando gran variedad de tecnologías de redes. El código fuente de estas aplicaciones es portable sin tener en cuenta el entorno de instalación. Como cualquier tecnología, ICE tiene su propio vocabulario. En ICE, los términos cliente y servidor más que designar partes concretas de una aplicación, denotan los papeles que son interpretados por partes de una aplicación durante una petición. Los clientes son entidades activas que emiten peticiones de servicio a los servidores. Los servidores son entidades pasivas que proveen servicios en respuesta a las peticiones (29).

[3]ObjectManagementGroup,consorcio dedicado al cuidado y el establecimiento de diversos estándares de tecnologías orientadas a objetos.

Un objeto ICE es una entidad conceptual o una abstracción. Cada objeto ICE tiene una interfaz con un determinado número de operaciones. Las interfaces, las operaciones y los tipos de datos intercambiados entre el cliente y el servidor se definen utilizando el lenguaje Slice. Este lenguaje permite definir el contrato entre el cliente y el servidor, independientemente del lenguaje de programación empleado. ICE proporciona un protocolo RPC que puede utilizar TCP/IP como capa de transporte subyacente. Además, ICE permite utilizar SSL, de forma que todas las comunicaciones entre el cliente y el servidor estén encriptadas (30).

El protocolo ICE define (30):

- Un determinado número de tipos de mensajes, como por ejemplo de solicitud y respuesta.
- Una máquina de estados que determina qué secuencia siguen los distintos tipos de mensajes que se intercambian el cliente y el servidor, junto con el establecimiento de conexión asociado.
- Reglas de codificación que determinan cómo se representa cada tipo de datos en la red.
- Una cabecera para cada tipo de mensaje que contiene detalles como el tipo del mensaje, el tamaño del mensaje y el protocolo y la versión de codificación empleados.

ICE soporta compresión en la red: Ajustando un parámetro de configuración se pueden comprimir todos los datos asociados al tráfico de red para reducir el ancho de banda utilizado. Esta propiedad resulta útil si la aplicación intercambia grandes cantidades de datos entre el cliente y el servidor. El protocolo ICE también soporta operaciones bidireccionales: Si un servidor quiere enviar un mensaje a un objeto de retrollamada proporcionado por el cliente, la retrollamada puede efectuarse a partir de la conexión establecida inicialmente por el cliente. Esta característica es especialmente importante cuando el cliente está detrás de un cortafuego que permite conexiones de salida pero no de entrada (30).

ICE maneja una serie de servicios, los cuales se implementan como servidores ICE de forma que la aplicación desarrollada actúe como un cliente. Ninguno de estos servicios utilizan características internas a ICE ocultas a la aplicación en cuestión, por lo que en teoría es posible desarrollar servicios equivalentes por parte del desarrollador. Sin embargo, manteniendo estos servicios disponibles como parte de la plataforma

permite al desarrollador centrarse en el desarrollo de la aplicación en lugar de construir la infraestructura necesaria en primer lugar. Dentro de los servicios que maneja ICE se pueden encontrar IceGrid, IceBox, IceStorm, IcePatch2 y Glacier2 (30).

Los middleware, provocan un alto consumo de recursos, la mayoría se distribuye bajo licencia GPL, presentan un costo de desarrollo y de implementación. Requieren ser instalado en cada equipo y algunos tienen una capacidad limitada de clientes. Es posible proveer interoperabilidad entre aplicaciones con otras tecnologías como DCOM4, sin embargo estas tecnologías al igual que CORBA, por lo general omiten o no soportan algunos de los principios de orientación a servicios. Por ejemplo, no presentan descubrimiento de servicios, pero puede ser una buena opción cuando se necesitan potenciar otros atributos como el tiempo de respuesta.

Se considera ICE como uno de los middleware más completos que permite su utilización en sistemas heterogéneos y consiente el desarrollo de aplicaciones distribuidas. Evita complejidades innecesarias, es fácil de aprender y utilizar, contiene herramientas y bibliotecas que facilitan el trabajo orientado a objetos. Permite crear y comunicar clientes y servidores, en diferentes lenguajes, plataformas, arquitecturas. Proporciona una implementación eficiente en carga de CPU, en ancho de banda y en uso de memoria. Aporta además una implementación basada en la seguridad, de forma que se pueda usar sobre redes no seguras.

1.2.2. Descripción actual del dominio del problema

La Universidad de Ciencias Informáticas, cuenta con diversos centros dedicados a la producción e investigación de software. En la facultad seis, se encuentra el centro GEySED, que se dedica a desarrollar productos y brindar soluciones en el campo de las señales digitales y la geoinformática. En este centro, los estudiantes ayudan en el desarrollo de software mediante la producción en los laboratorios de proyecto. Con el inicio del curso académico 2011-2012 en el departamento docente - productivo de Señales Digitales del centro GEySED, se integraron los proyectos PTARTV, Primicia y Video Web de donde surgió el proyecto Sistema de gestión y transmisión de contenidos audiovisuales (SIAV).

SIAV no es más que un sistema capaz de automatizar procesos relacionados a la gestión, procesamiento y transmisión de contenidos multimedia. El sistema está

[4]DistributedComponentObjectModel, tecnología para desarrollar software distribuido,

compuesto por varias plataformas y subsistemas que pueden desplegarse independientemente o integrados en un mismo sistema, lo que provee facilidades para la personalización atendiendo a los requisitos planteados por los clientes. SIAV contiene actualmente distintas plataformas funcionando, cada una integrada de diversos subsistemas, entre estas se encuentran:

Plataforma SIAV Web Tv:

- Subsistema de administración.
- Subsistema de presentación.
- Subsistema componente gestor de procesos de media.

Plataforma SIAV Noticias:

- Subsistema de administración.
- Subsistema de transmisión.

Plataforma SIAV Transmisión:

- Subsistema de monitorización.
- Subsistema de transmisión.
- Subsistema de programación.

1.2.3. Situación problemática

EnSIAVexistecomodeficiencia,quelossubsistemasyaplicacionesdelosproductosquedieron pasoasusurgimiento,nopresentanunabuenainteroperabilidad. Lograr interoperabilidad entre las aplicacionesseríamuyfactibleparaquelosprocesossecomplementaranentresí. De esta manera se podrían aprovechar las funcionesexistentesencadaproducto.Por tal motivo sehacenecesariorealizarunestudiodetalladosobrelosmétodosytécnicasautilizarparalograrquetodaslasaplicacionesdel sistemapuedaninteractuarentresí.

Esto se debe a que actualmente en SIAV la integración entre aplicaciones se realiza principalmente, mediante el uso de la base de datos. Para que una aplicación utilice funcionalidades de otra, debe realizar la petición de esta a través de consultas a la base de datos. En caso de agregar nuevos subsistemas se tendrá que reelaborar todo el modelo de datos de modo tal que se tenga toda la información centralizada en una base de datos común. Esto provoca el trabajo con una base de datos demasiado extensa y que se dificulte además la comunicación entre las aplicaciones. Por todas

estas cuestiones se tiene un producto poco escalable, existe dependencia entre las aplicaciones y no se cuenta con una buena integración entre ellas.

En busca de eliminar estas insuficiencias se tiene planificado alcanzar una solución orientada a servicios, capazdeoptimizarlacomunicación,integraciónycooperaciónentrelasaplicacionesdeSIAV atravésdeservicios,peroqueasuvezlossistemasseanindependientes. Se quiere que los sistemas puedan acceder entre ellos con la simple llamada de una función, dondeseespecifiquenlosdatosdeentradaylosdesalida.Conestonointeresadondeseencuentralainformaciónnicómoserealizalaobtencióndeesta.Asísiunodeestossubsistemasdejaradefuncionar,osequiereconectarconotro,nosufredaño la comunicación. Para esto el intercambio de datos debe realizarse mediante estándares de comunicación bien establecidos. De esta manera se brindará más dinamismo e interoperabilidad entre los subsistemas de SIAV.

1.3. Factores de seguridad y comunicación entre aplicaciones

Uno de los elementos clave que afectan el diseño de un sistema distribuido, es el modo de diseñar la infraestructura de comunicación para cada parte de la solicitud. Los componentes deben comunicarse entre sí.

Para que dos programas puedan comunicarse entre sí es necesario que se cumplan ciertos requisitos (17):

- Que un programa sea capaz de localizar al otro.
- Que ambos programas sean capaces de intercambiarse cualquier secuencia de datos relevantes a su finalidad.

Existen dos tipos de comunicaciones, síncrona y asíncrona. La comunicación síncrona se adapta mejor a escenarios en los que hay que, garantizar el orden en que las llamadas son recibidas, o cuando se debe esperar a que la llamada vuelva. La comunicación asincrónica se adapta mejor a escenarios en los que la capacidad de respuesta es importante. La comunicación basada en mensajes permite brindar servicios a clientes. Para esto se debe tener en cuenta que una conexión no siempre estará presente y puede que los mensajes necesiten ser almacenados y enviados

cuando una conexión esté o no disponible. Se debe considerar el caso de que la respuesta de un mensaje no se reciba, para esto se podrían registrar los mensajes enviados, para su posterior procesamiento en caso de no recibir respuesta. Si se logra la comunicación utilizando métodos que imponen un conjunto mínimo de requisitos, se tendrá una aplicación débilmente acoplada. Para un buen rendimiento y prestación del servicio, se debe evitar pasar datos innecesarios a los métodos remotos y reducir al mínimo el volumen de datos enviados a través de la red. Esto reduce la carga de la serialización y la latencia de la red (17).

Una estrategia de comunicación segura podría ser, proteger los datos sensibles de ser leídos cuando son transmitidos sobre la red y de ser manipulados y si es necesario, garantizar la identidad de la persona que envía el mensaje. Existen dos tipos de seguridad fundamentales en las comunicaciones: la seguridad de transporte y la seguridad de mensajes. Para disfrutar de la máxima protección, se podría tener en cuenta la combinación de técnicas de seguridad de transporte y de mensajes (17):

Seguridad de Transporte:
La seguridad de transporte se utiliza para proporcionar punto a punto, la seguridad entre los dos extremos. Proteger el canal evita que los atacantes tengan acceso a todos los mensajes del canal. Los enfoques comunes para la seguridad de transporte, son Secure Sockets Layer (SSL) y Seguridad del protocolo Internet (IPSec). Si la interacción entre el servidor y el consumidor no se enrutan a través de intermediarios, se puede utilizar la seguridad de transporte. La seguridad de transporte es una buena opción para asegurar la comunicación entre un cliente y un servidor situado en una red privada, como una intranet(17).

Seguridad de mensajes:
La seguridad de mensajes, se puede utilizar con cualquier protocolo de transporte. Se debe proteger individualmente el contenido de los mensajes, cada vez que pasa por una red no segura. La seguridad de mensajes se puede implementar para los mensajes sensibles que pasan fuera de la seguridad de la red. Si hay intermediarios entre el cliente y el servidor, utilice la seguridad de mensajes para los mensajes sensibles, ya que garantiza de extremo a extremo la seguridad. El intermediario terminará la conexión SSL o IPSec cuando reciba el mensaje y luego crea una nueva conexión SSL o IPSec para pasar al servidor. Por lo tanto, si no se utiliza la seguridad de mensajes, en este caso, se pueden tener acceso a los mensajes, por medio del

intermediario(17).

1.4. Definición de modelo de comunicación

Un modelo es, básicamente, una abstracción que incluye loesencial de un problema para el receptor del modelo. Filtra los detalles no esenciales, de forma queel problema se hace más comprensible. Existen varios tipos de modelos, la realización del presente trabajo se basa en un modelo científico.

Modelo científico:
Resultado de generar una representaciónabstracta, conceptual, gráfica o visual. Este permite analizar,describir, explicar, simular o predecir fenómenos y procesos(35).

Para la elaboración del modelo de comunicación se deben tener en cuenta los siguientes requisitos:

- Especificar estructura del modelo.
- Visualizar alcance del modelo.
- Especificar condiciones necesarias para su aplicación.

El modelo de comunicación dinámica presenta varios objetivos, los cuales son necesarios para entender el porqué de su aplicación. Estos objetivos brindan una guía para la elaboración y entendimiento del modelo.

- Describir el modelo.
- Detallar funcionamiento del modelo.
- Generar documentación que justifique su importancia.

1.5. Arquitectura de software

La arquitectura es una vista estructural de alto nivel. Define estilo o combinación de estilos para una solución. Se concentra en requisitos no funcionales. Los requisitos funcionales se satisfacen mediante modelado y diseño de aplicación. Es esencial para el éxito o fracaso de un proyecto (36).

La arquitectura de software permite representar la estructura de un sistema a un nivel mayor que el dado por la programación e incluso por el diseño. Para que la

arquitectura se convierta en una herramienta útil dentro del desarrollo y mantenimiento de los sistemas de software es necesario que se cuente con una manera precisa de representarla. La arquitectura de software constituye una vista del sistema que incluye los componentes principales según se le percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión del detalle inherente a la mayor parte de las abstracciones(37).

Patrones

En principio, se puede pensar en un patrón como una manera especialmente inteligente e intuitiva de resolver una clase de problema en particular. El patrón es una descripción del problema y la esencia de su solución, de forma que la solución se pueda reutilizar en diferentes situaciones, es una solución adecuada a un problema común. Los patrones se definen por un nombre, un problema, una solución y las consecuencias de su aplicación. En dependencia del problema que solucionan, estos pueden ser clasificados en: Patrones de arquitectura, patrones de análisis, patrones de diseño, entre otros. Los patrones arquitectónicos expresan el esquema de organización estructural fundamental para sistemas de software, proveen un conjunto de subsistemas predefinidos, especifican sus responsabilidades e incluyen reglas y pautas para la organización de las relaciones entre ellos(38).

Patrones Arquitectónicos

Un estilo arquitectónico, a veces denominado patrón arquitectónico, es un conjunto de principios que proporciona un marco abstracto para una familia de sistemas. Un estilo arquitectónico mejora la separación y promueve la reutilización del diseño, aportando soluciones a los problemas que se repiten con frecuencia. Un estilo arquitectónico determina el vocabulario de los componentes y conectores que se pueden utilizar en casos de ese estilo, junto con un conjunto de restricciones sobre cómo pueden ser combinados. Estos pueden incluir restricciones topológicas en las descripciones arquitectónicas (por ejemplo, sin ciclos) (39).

Estilos de llamada y retornos:

- Modelo vista controlador
- Arquitectura en capas
- Arquitectura orientada a objetos

- Arquitectura basada en componentes

Estilos punto a punto:

- Arquitectura basada en eventos
- Arquitectura orientada a servicios
- Arquitectura basada en recursos

Estilos de flujo de datos:

- Tubería y filtros

Luego de un análisis previo a la arquitectura de software y a los patrones arquitectónicos, se determinó que era factible, por ser una arquitectura modular y jerárquica, aplicar unaarquitectura basada en componentes y una arquitectura orientada a objetos, debido a que permite aprovechar las ventajas de la programación orientada a objetos, dentro de las que se encuentran abstracción y polimorfismo.

Arquitectura basada en componentes

La arquitectura basada en componentes, consiste en una rama de la Ingeniería de Software en la cual, se trata con énfasis la descomposición en componentes funcionales. Esta descomposición permite convertir, componentes del software pre-existentes en piezas más grandes de software. Este proceso de construcción de una pieza de software con componentes ya existentes, da origen al principio de reutilización del software, mediante el cual se promueve que los componentes sean implementados de una forma que permita su utilización funcional sobre diferentes sistemas en el futuro(40).

Los siguientes son los principales beneficios del estilo de arquitectura basado en componentes(17):

- Facilidad de despliegue: Cuando una nueva versión esté disponible, usted podrá reemplazar la versión existente sin impacto en otros componentes o el sistema como un todo.
- Reducción de costos: El uso de componentes de terceros permite distribuir el costo del desarrollo y del mantenimiento.
- Facilidad de desarrollo. Los componentes implementan una interfaz bien definida para proveer la funcionalidad definida permitiendo el desarrollo sin

impactar otras partes del sistema.

- Reusable. El uso de componentes reutilizables significa que ellos pueden ser usados para distribuir el desarrollo y el mantenimiento entre múltiples aplicaciones y sistemas.
- Mitigación de complejidad técnica. Los componentes mitigan la complejidad por medio del uso de contenedores de componentes y sus servicios. Ejemplos de servicios de componentes incluyen activación de componentes, gestión de la vida de los componentes, gestión de colas de mensajes para métodos del componente y transacciones.

La estructura de la arquitectura basada en componentes contempla 3 partes(17):

- El nombre de los componentes: El nombre de un componente debe ser la identificación de la funcionalidad y uso que tiene como software. Generalmente, los desarrolladores usan algún tipo de convención que facilite la identificación de componentes, especialmente, cuando se trabaja en proyectos de gran envergadura.
- La interfaz de los componentes: Es el área de intercambio (input-output) entre el interior y el exterior de un componente de software. La interfaz es quien permite acceder a los datos y funcionalidades que estén especificadas en el interior del componente (acceder funcionalmente, no a su especificación). Adicional a la interfaz se encuentra la documentación que muestra la información sobre cómo utilizar un componente.
- Cuerpo y código de implementación: Es la parte del componente que provee la forma (implementación) sobre la cual un fragmento del componente realiza sus servicios y funcionalidades. Este es el área que debe cumplir con el principio de encapsulación.

La reusabilidad es una de las características más importantes en el desarrollo de sistemas bajo una arquitectura basada en componentes. Un componente de software debe ser diseñado de tal manera que pueda ser reutilizado en otros sistemas. Este principio de reutilización del componente, requiere un esfuerzo extra por el equipo de desarrollo que se basa en(17):

- Una documentación completa de cada atributo y funcionalidad del componente.
- Una etapa de pruebas organizada y certera que certifique el correcto

funcionamiento del componente.

- Una definición de comprobaciones precisa para el chequeo de cada parámetro de entrada (input) del componente.
- Un manejo de notificaciones de errores preciso, que advierta de la existencia de estos de una forma apropiada.
- Desarrollar teniendo en cuenta que el componente puede ser requerido para trabajar en muchos contextos muy diferentes unos de otros (tomar en cuenta la eficiencia, uso de memoria y recursos).

ComoModelo Arquitectónico, se seleccionó Cliente/Servidor, pues describe la arquitectura de sistemas distribuidos, que involucran un cliente independiente, un sistema de servidor y una red de conexión.

Modelo Cliente/Servidor

Desde el punto de vista funcional, se puede definir la computación Cliente/Servidor como una arquitectura distribuida que permite a los usuarios finales obtener acceso a la información en forma transparente aún en entornos multiplataforma. En el modelo cliente servidor, el cliente envía un mensaje solicitando un determinado servicio a un servidor y este envía uno o varios mensajes con la respuesta. En un sistema distribuido cada máquina puede cumplir el rol de servidor para algunas tareas y el rol de cliente para otras. La arquitectura Cliente/Servidor es una extensión de programación modular en la que la base fundamental es separar una gran pieza de software en módulos con el fin de hacer más fácil el desarrollo y mejorar su mantenimiento (41).

El Cliente normalmente maneja todas las funciones relacionadas con la manipulación y despliegue de datos, por lo que están desarrollados sobre plataformas que permiten construir interfaces gráficas de usuario (GUI), además de acceder a los servicios distribuidos en cualquier parte de una red. El servidor es el proceso encargado de atender a múltiples clientes que hacen peticiones de algún recurso administrado por él. El servidor normalmente maneja todas las funciones relacionadas con la mayoría de las reglas del negocio y los recursos de datos. En la figura 4, se observa el proceso donde, el cliente envía el requisito al servidor, este lo procesa, envía el resultado y finalmente el cliente lee el resultado(41).

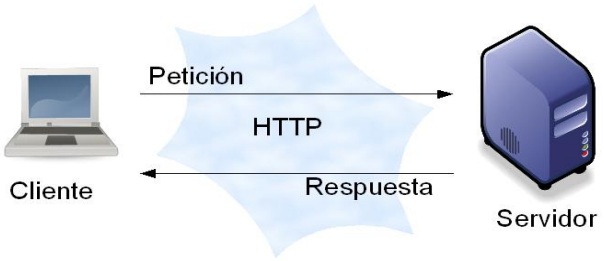


Figura 4. Modelo Cliente/ Servidor.

Las características básicas de una arquitectura Cliente/Servidor son(21):

- Combinación de un cliente que interactúa con el usuario y un servidor que interactúa con los recursos compartidos. El proceso del cliente proporciona la interfaz entre el usuario y el resto del sistema. El proceso del servidor actúa como un motor de software que maneja recursos compartidos tales como bases de datos, impresoras, módems, etc.
- Se establece una relación entre procesos distintos, los cuales pueden ser ejecutados en la misma máquina o en máquinas diferentes distribuidas a lo largo de la red.
- Existe una clara distinción de funciones basada en el concepto de servicio, que se establece entre clientes y servidores.
- La relación establecida puede ser de muchos a uno, en la que un servidor puede dar servicio a muchos clientes, regulando su acceso a recursos compartidos.
- Los clientes corresponden a procesos activos en cuanto a que son éstos los que hacen peticiones de servicios a los servidores. Estos últimos tienen un carácter pasivo ya que esperan las peticiones de los clientes.
- No existe otra relación entre clientes y servidores que no sea la que se establece a través del intercambio de mensajes entre ambos. El mensaje es el mecanismo para la petición y entrega de solicitudes de servicio.
- El ambiente es heterogéneo. La plataforma de hardware y el sistema operativo del cliente y del servidor no son siempre la misma. Precisamente una de las principales ventajas de esta arquitectura es la posibilidad de conectar clientes y servidores independientemente de sus plataformas.

Otras variaciones en el estilo de cliente / servidor son(17):

- Sistemas Cliente-cola-cliente: Este enfoque permite a los clientes, comunicarse con otros clientes a través de una cola ubicada en el servidor. Los clientes pueden leer datos y enviar datos a un servidor que actúa simplemente almacenando los datos como una cola. Esto permite a los clientes distribuir y sincronizar archivos e información. Esto a veces se conoce como una arquitectura de cola pasiva.
- Aplicaciones punto a punto (P2P): Desarrollado a partir del estilo cliente-Cola-Cliente, el estilo P2P permite que el cliente y el servidor puedan intercambiar sus papeles para distribuir y sincronizar archivos e información entre varios clientes. Se extiende el estilo cliente / servidor a través de múltiples respuestas a las solicitudes, los datos compartidos, el descubrimiento de recursos y la resistencia a la eliminación de sus compañeros.
- Los servidores de aplicaciones: Un estilo de arquitectura especializada en los hosts del servidor y ejecuta las aplicaciones y servicios, a las que un cliente ligero accede a través de un navegador o cliente especializado instalado en el software.

La utilización del modelo cliente/servidor, en la presente propuesta, ofrece algunos beneficios como la facilidad de mantenimiento, se distribuyen entre varios servidores que se conocen el uno al otro a través de una red. Esto asegura que un cliente permanece inconsciente y no se vea afectado por la reparación, actualización o reubicación del servidor.

Patrones de diseño

Se pueden definir como: Soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia y que se ha demostrado que funcionan. Los patrones de diseño no son fáciles de entender, pero una vez entendido su funcionamiento, los diseños serán mucho más flexibles, modulares y reutilizables. Los patrones de diseño reflejan todo el rediseño y remodificación que los desarrolladores han ido haciendo a medida que intentaban conseguir mayor reutilización y flexibilidad en su software(42).

Un patrón de diseño nomina, abstrae e identifica los aspectos clave de una estructura de diseño común, lo que los hace útiles para crear un diseño orientado a objetos reutilizable. El patrón de diseño identifica las clases e instancias participantes, sus roles y colaboraciones y la distribución de responsabilidades. Cada patrón de diseño

se centra en un problema concreto, describiendo cuándo aplicarlo y si tiene sentido hacerlo teniendo en cuenta otras restricciones de diseño, así como las consecuencias, ventajas e inconvenientes de su uso(43).

En el presente trabajo se utiliza un patrón de comportamiento, el observer, a continuación se detallan sus características.

Patrones de Comportamiento

Tienen que ver con algoritmos y asignación de responsabilidades. Estos patrones se focalizan en el flujo de control dentro de un sistema. Ciertas formas de organizar los controles dentro del sistema pueden llevar a grandes beneficios en cuanto a mantenibilidad y eficiencia. Algunos ejemplos de estos patrones incluyen la definición de abstracciones de algoritmos, las colaboraciones entre objetos para realizar tareas complejas reduciendo las dependencias o asociar comportamiento a objetos e invocar su ejecución(44).

Observer

Brinda un mecanismo, que permite a un componente transmitir de forma flexible mensajes a aquellos objetos que hayan expresado interés en él. Estos mensajes se disparan cuando el objeto ha sido actualizado y la idea es que quienes hayan expresado interés reaccionen ante este evento. Define una dependencia de uno a muchos entre objetos, de forma que cuando un objeto cambia de estado se notifica y actualizan automáticamente todos los objetos(45).

1.6. Análisis de otras soluciones existentes

Acontinuación,seexponeninvestigacionessobresolucionesexistentesenlaactualidadrelacionadas alpresentetema.

2mdc.com

2mdc.com es una compañía experta en el desarrollo de **aplicaciones web** y contenidos específicos para su difusión por Internet. Entre sus funcionalidades principales, brindan servicios web, mediante el protocolo de comunicación SOAP, que permiten el acceso seguro, desde el exterior y de todo tipo de usuarios (registrados o no), a determinados servicios(46).

MPM Tedial

MPM es un middleware, utilizado en la compañía Tedial para comunicar sistemas heterogéneos. MPM es una aplicación que permite automatizar flujos de trabajo, trabajando como una capa intermedia. Asegura la integración de todos los sistemas de terceros y tecnologías de una instalación típica heterogénea como servidores de vídeo, automatización de emisión, editores, el sistema de sala de redacción, los dispositivos de subtitulado o control de calidad automático. MPM no sólo asegura que los medios de comunicación y el flujo de metadatos sean libres entre los sistemas de terceras partes de la instalación. Permite que sea posible llevar a cabo flujos de trabajo de producción que abarcan toda la instalación, a pesar de que muchos de los sistemas implicados son a menudo incompatibles ya sea por razones técnicas o comerciales(47).

Las soluciones existentes anteriormente planteadas, exponen mecanismos utilizados para brindar servicios, aportando interoperabilidad a la compañía. Técnicas como MPM Tedial no se podría utilizar ya que es un software propietario, desarrollado específicamente para las aplicaciones de la compañía Tedial.

1.7. Conclusiones parciales

Enestecapítulosepuntualizarontodosloselementosimportantesentornoalas técnicas yprocesosdeinteroperabilidad entre aplicaciones y componentes.Se caracterizaron soluciones que garantizan interoperabilidad entre sus componentes de software. Se definió el concepto de modelo. Se observa que existen numerosas tecnologías para comunicar aplicaciones, pero luego de un estudio se asimila que, cualquiera no sería factible para dar solución al presente problema. Se sentaron las bases para la realización del modelo de comunicación, de forma que permita brindar interoperabilidad entre aplicaciones.

CAPÍTULO 2: Descripción de la propuesta de solución

En SIAV, se quiere que cada subsistema tenga la posibilidad de acceder a cualquier funcionalidad, presentando mayor flexibilidad y una buena interoperabilidad entre las aplicaciones. Cada subsistema podría publicar sus funcionalidades como servicios, para que todos tengan acceso a ellas satisfaciendo el problema existente. Sin embargo esta solución es relativamente simple e ideal cuando el número de aplicaciones que tienen que intercambiar datos es muy reducido y estático. Presentando como desventaja, que cada vez que se quieran agregar nuevos servicios, hay que realizarles modificaciones al código fuente. Se debe avisar a las demás aplicaciones de que existe un nuevo servicio y especificar sus características. Cada una de estas aplicaciones debe entrar al código fuente e implementar el consumo del nuevo servicio. Esta solución se vería atada a una tecnología, por lo que si se quisiera cambiar la tecnología por cualquier motivo, ya esta solución no sería de utilidad.

Para resolver el problema de investigación, se establece un modelo de comunicación que brinda dinamismo y escalabilidad. En este capítulo se describe la solución propuesta, que consiste en la descripción de un modelo de comunicación entre aplicaciones. Este modelo tiene como objetivo establecer una estructura y un modo de funcionamiento para obtener interoperabilidad en SIAV.

La utilización de estándares y servicios web brinda óptimos resultados. En lugar de tratar con múltiples aplicaciones incompatibles en varios equipos y lenguajes de programación, es posible agregar una capa de abstracción, basada en estándares y fácil de integrar con prácticamente cualquier entorno. De esta manera se obtienen aplicaciones más flexibles y se refuerza la interoperabilidad entre estas.

2.1. Descripción general del proceso

2.1.1. Estructura del modelo

En la modelación del procedimiento, las palabras que se utilizaron fundamentalmente para describir sus elementos fueron:
Servicios: representan las funcionalidades a las que se quiere acceder.
Cliente: realiza una petición.
Servidor: brinda respuesta a las peticiones de los clientes.
Capa: porción del código aislado de la lógica de programación.

2.1.2. Alcance del modelo

Este modelo puede ser ajustado y adaptado a cualquier iniciativa de un proyecto de la UCI o empresa que desee aplicar el modelo para lograr interoperabilidad y dinamismo entre sus aplicaciones. Brinda la representación y guía objetiva para lograr y llevar a cabo los procesos implicados en la aplicación del modelo, dentro de cualquier sistema que se aliste a implantar el modelo de comunicación dinámica. La implantación de políticas para el despliegue y producto final del proyecto quedan fuera del alcance del modelo.

2.1.3. Condiciones necesarias para aplicar el modelo

Cada aplicación, debe agregar a su implementación el modelo de comunicación. El servidor debe integrar varias tecnologías y el cliente solo necesita conocer su tecnología.Cliente y servidor se deben comunicar usando la misma tecnología, en caso contrariose enviará un error como resultado. El cliente debe conocer la dirección del servidor por el cual se publica el servicio, así como el puerto que escucha las peticiones de la tecnología del cliente.

2.1.4. Funcionalidades a publicar

Como se menciona anteriormente SIAV está compuesto por varios subsistemas. Cada uno de los subsistemas por los que está compuesto SIAV presenta numerosas funcionalidades, que podrían ser de necesidad común entre todos.
Subsistema SIAV Web Tv:
Adicionar un nodo a las listas de usuario personalizadas.
Eliminar un nodo de las listas de usuario personalizadas.
Retornar las listas personalizadas del usuario que este logueado.
Adicionar una lista.
Cambiar el nombre de una lista personalizada.
Eliminar una lista personalizada.
Devolver el voto de un nodo.
Cambiar el voto de un nodo.
Obtener la información de un nodo.
Buscar los nodos que contengan una palabra clave.
Salvar los cambios realizados en una sección.
Obtener el árbol de las Categorías de una sección.
Loguear un usuario.
Desloguear el usuario que esta logueado.
Devolver las medias más vistas.
Devolver las medias destacadas.
Devolver si un usuario tiene un permiso determinado.
Devolver las medias que fueron publicadas más recientemente.
Devolver todas las secciones.
Devolver los datos de una media.
Devolver todos las contenidos multimedias y los artículos de contenidos disponibles en la plataforma, con el estado publicado o no publicado de cada uno.
Cambia el estatus (publicado) de un contenido.
Subsistema Transmisión:
Obtener la programación del día.
Subsistema SIAV Noticias:
Obtener noticias del día.
Obtener clima.

2.2. Descripción del modelo

El modelo establece la publicación de un servicio que permite que las aplicaciones

invoquen métodos y se comuniquen entre sí. Presenta una capa de comunicación, donde se separa todo el código referente a las tecnologías. Establece una capa de publicación y una capa de consumo, que serán integrados según las necesidades de publicación o consumo del subsistema.
Está basado en el estilo arquitectónico cliente/servidor. Específicamente utiliza una de sus variaciones: punto a punto, donde las aplicaciones se vuelven híbridas, para llevar a cabo la función de cliente o de servidor, según quien envía o responde una petición.

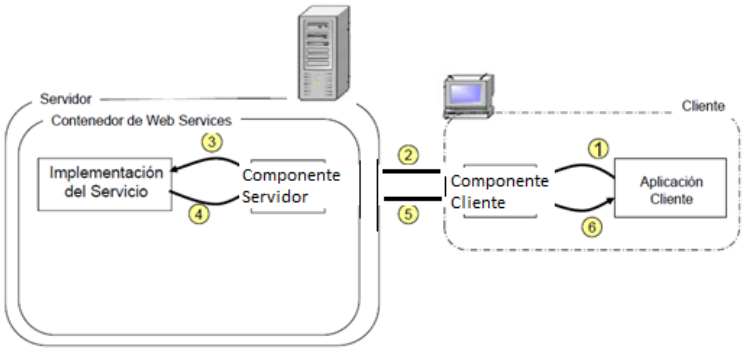


Figura 5. Comunicación cliente/servidor.
Este modelo garantiza el trabajo con eventos, basado en el propio lenguaje de programación orientado a objetos. En la programación orientada a objetos los eventos se apoyan en parámetros y permiten ejecutar funciones creadas por los desarrolladores, utilizando los parámetros correspondientes a cada evento para realizar la acción deseada.

2.2.1. Capa de publicación

Se compone de un conjunto de elementos que brindan información y funcionalidades al usuario del sistema. Este presenta un marco visual estándar para los subsistemas de SIAV.
Su utilidad está orientada a publicar funcionalidades como servicios. Implementa un mecanismo dinámico que permite, dar respuesta a las peticiones de los clientes. Para esto se le debe pasar el nombre de una funcionalidad y una lista con los parámetros que esta requiere, los cuales parsea posteriormente. Los servicios se publicaran a través de un puerto determinado, por el cual se estará escuchando las peticiones de los clientes. Los parámetros, serán un convenio entre el cliente y el servidor.
Elementos que definen lacapa de publicación:
Nombre del servicio.
Contenedor donde se muestran las peticiones respondidas, notificaciones y errores.
Puerto.
Botón para activar servicio.

2.2.2. Capa de consumo

Es unacapa orientada a facilitar el consumo de los servicios. Se incluye por defecto campos para especificar el nombre de la funcionalidad y los parámetros de conexión.
El comportamiento general de estacapa consiste en enviar al servidor los datos de los campos del formulario, los cuales constituyen los criterios de la búsqueda a realizar. Se debe mostrar el resultado de la petición en el cuadro definido por el desarrollador. Para la realización de la búsqueda, se genera un botón que ejecuta la acción de enviar los datos al servidor.
Posee un área de información donde se muestra el estado de la operación y cuando ocurre algún error durante la transferencia, se le informa al usuario y se cancela la acción. Generalmente los errores que se presentan son por falta de conexión al

servidor.
Elementos que definen la capa de consumo:
Nombre de la funcionalidad a invocar servicio.
Puerto.
Ip del servidor.
Botón para solicitar servicio.

Figura 6. Especificaciones de capas.

2.2.3. Diversidad de tecnologías

Cada tecnología tiene sus ventajas y desventajas y brindan diferentes funcionalidades. Se quiere implementar una capa de comunicación dinámica, que no esté atada a una tecnología en específico, sino que use la que necesite, en dependencia de las prestaciones que sean necesarias para SIAV en ese momento.
Motivos por lo cual sería bueno usar varias tecnologías: basados en tecnologías middleware, se logra estandarizar, gestionar y controlar el intercambio de información entre los componentes de software. Además cuentan con interfaces integradas y fuentes de datos comunes. Mediante una tecnología, basada en estándares como puede ser XML-RPC, se logran comunicaciones asíncronas, de manera sencilla; o utilizando SOAP se pueden describir y publicar los servicios disponibles.
Por tanto sería factible lograr una solución donde, la tecnología a utilizar sea en dependencia de las necesidades del sistema.

2.2.4. Utilización de Plug-in

La implementación del modelo, establece la integración de un plug-in, donde se encuentra el código relacionado con la tecnología, facilitando el trabajo con las mismas. Al tener la posibilidad de separar el código de la tecnología, de la lógica del negocio, no se está atado a una tecnología específica, brindando más dinamismo al modelo. Todos los plug-in se representarían mediante la misma estructura.
Otro aspecto a tener en cuenta es que, las tecnologías se comercializan bajo licencia GPL, esto las convierteen tecnologías libres cuando se desarrollan basadas en plug-in. Si fuera necesario liberar el código, solo se libera el código del plug-in, de esta forma no se afecta el desarrollo de la aplicación.

2.2.5. Aspectos de comunicación y seguridad

En las comunicaciones es primordial contar con una buena estabilidad, reutilización y rendimiento. Por tanto se cuenta con una comunicación asincrónica, permitiendo de esta manera que no se detenga el flujo de llamadas, en caso de que, quien envíe un mensaje no obtenga respuesta del receptor.En el presente trabajo la seguridad es aplicada desde las tecnologías. Se seleccionaron tecnologías que brindan opciones de seguridad.El modelo propuesto define una estructura totalmente desacoplada ya que usa una arquitectura basada en plug-ins.

2.3. Funcionamiento del modelo

La aplicación servidora, publicará un único servicio, el cual se encargará de llamar a la funcionalidad requerida por el cliente. Las aplicaciones servidoras, tendrán unacapa de publicación y las aplicaciones clientes, unacapa de consumo. Todo el código de la tecnología estará elaborado en un plug-in, dentro de la capa. De esta manera al necesitar cambiar de tecnología, solo se cambia el plug-in, sin necesidad de cambiar el resto del código.
El consumidor enviará una petición al servidor, con un ID de petición. Esto es necesario, ya que el servidor recibirá varias peticiones simultáneas y necesitará un modo de organizar sus resultados y así enviar cada respuesta a la petición correcta. Esta petición es de tipo genérica, ya que se desconoce la funcionalidad a la que se quiere acceder. En este envío se especifican los parámetros convenidos con el servidor, donde el primer parámetro será el nombre de la funcionalidad a la que se quiere acceder. El servidor recibe la petición, como se muestra en la figura 7, toma el nombre de la funcionalidad y busca si existe. Estas funcionalidades están implementadas mediante condiciones. Al comprobar que la funcionalidad existe, se ejecuta y finalmente devuelve una respuesta o un error.

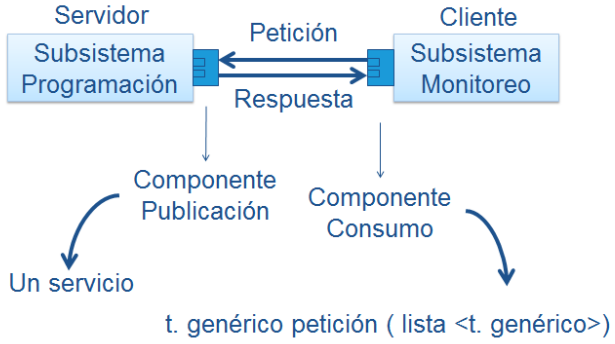


Figura 7. Representación del modelo.

2.4. Ventajas del modelo de comunicación

El modelo, establece un margen a seguir por los desarrolladores, lo cual facilita que estos se acojan a dicho estándar y les sea más comprensible a futuros desarrolladores para su optimización y modificación.A continuación se relacionan las ventajas del modelo planteado.
Reutiliza las funcionalidades de los subsistemas que se encuentran en SIAV. Dicho de otra manera quiere decir que cuando una aplicación requiera utilizar las funcionalidades ofrecidas por algún subsistema de SIAV, solo tendrá que consumir el servicio definido por el modelo. De esta manera, el implementador no tiene que preocuparse de la lógica ni de la implementación de las aplicaciones.
Se pueden implementar funcionalidades que den soluciones a problemas existentes en el lenguaje que más fácil o conveniente sea, debido a que estas funcionalidades pueden ser consumidas accediendo al servicio definido por el modelo.
Provee al programador de una abstracción total de las funcionalidades usadas.
Los cambios o actualizaciones de los servicios no suponen problema para los usuarios porque este proceso es transparente para ellos.
Desarrollo de aplicaciones más rápido y económico.

2.5. Conclusiones parciales

Mediante la confección de este capítulo, se da cumplimiento al objetivo relacionado con la creación del modelo.Se ha realizado la descripción, estructura, alcance y condiciones para su aplicación, así como los elementos que garantizan su funcionamiento, además de su presentación y la representación del funcionamiento de las capas de publicación y consumo.Se hicieron posible la comprensión de los pasos que se requieren para lograr la implementación de forma sencilla del modelo, además de como facilitar su desarrollo y ejecución.

CAPÍTULO 3: Prueba y validación del modelo propuesto

Enestecapítulosedescribe la fase de prueba y validación de la propuesta de solución. El objetivo de la fase de prueba al modelo propuesto, es detectar las anomalías que presenta y comprobar las necesidades que satisface. La validación del modelo propuesto, se lleva a cabo mediante el método Delphi.

3.1. Prototipo funcional

Se desarrolló un prototipo funcional para probar el funcionamiento del modelo propuesto. Como resultado del análisis realizado en el capítulo anterior a las diferentes tecnologías y atendiendo a los elementos más útiles encontrados en ellas, se tuvieron en cuenta ICE y XML-RPC, como las más factibles para integrar en el prototipo funcional.

Para eldesarrollo del prototipo funcional, se seleccionaron una serie de herramientas. La implementación se realizó con el lenguaje de programación C++, que permite la manipulación de objetos y presenta variada documentación y bibliotecas de funciones para distintas tecnologías. Como framework de desarrollo se utilizó Qt y como entorno integrado de desarrollo QtCreator. Se apoya en una arquitectura basada en componentes, ya que permite la escalabilidad necesaria en el modelo, así como, la reutilización de código.

Breve caracterización de las tecnologías a probar:
Se realizó un prototipo funcional, que integra tecnologías diferentes, mediante la implementación de plug-ins. Para poder realizar la propuesta de estas tecnologías en el sistema SIAV se verán a continuación características particulares que las definen:

ICE: Permite implementar sistemas punto a punto, brinda un conjunto de herramientas orientada a objetos que permite construir aplicaciones distribuidas con un mínimo de esfuerzo. Proporciona una implementación eficiente en ancho de banda, en el uso de memoria y de CPU. Además presenta seguridad incorporada, por lo que es adecuado para las redes del centro. ICE presenta APIs, herramientas y bibliotecas para el desarrollo de sistemas heterogéneos. El cliente y el servidor pueden ser desarrollados en diferentes lenguajes de programación y descritos en distintos sistemas operativos y arquitectura. Puede comunicarse además con una variedad de tecnologías de redes.

XML-RPC: Es una manera rápida y fácil de hacer llamadas a procedimientos remotos. Convierte la llamada a procedimiento en un documento XML, lo envía a un servidor remoto mediante HTTP y vuelve la respuesta como XML. Como desventaja se podría mencionar, que no tiene un lenguaje neutro para describir los servicios disponibles, además de que es necesario conocer de antemano los servicios existentes. Se puede utilizar con distintos lenguajes como php, c++, java y otros.

Al prototipo funcional se le realizaron pruebas para verificar su comportamiento bajo condiciones anormales. Para esto se crearon numerosos clientes consumiendo los servicios de un mismo servidor, mediante el uso de dos tecnologías diferentes. De esta manera se pudo apreciar que el modelo está capacitado para soportar una gran carga de peticiones. Se identificaron las respuestas del servidor, cuando el cliente no realiza lo que debe hacer o cuando los parámetros de conexión no son los correctos. Se enviara un error al cliente siempre que los puertos del cliente y servidor no coincidan; el cliente no conozca el ip del servidor; el servidor no tenga integrado la tecnología que usa el servidor. Se realizaron pruebas al prototipo funcional para verificar su rendimiento en cuanto a, tiempo de respuesta del servidor ante una petición. Se demostró que mediante la utilización de diversas tecnologías, existen diferencias en el tiempo de respuesta del prototipo funcional.

Con la realización del prototipo funcional se probó que el modelo propuesto satisface las necesidades de brindar interoperabilidad entre los subsistemas de SIAV. Se demostró que un servidor pude brindar servicios a diferentes clientes. Se comprobó que un servicio puede consumirse y publicarse mediante diferentes tecnologías, aportando dinamismo al producto SIAV.

3.2. Validación del modelo propuesto

Se hace necesario tener el criterio de otras personas, que presenten conocimiento necesario para saber si la investigación realizada está correcta. Para utilizar posteriormente el modelo propuesto en SIAV, es necesario tener validada dicha propuesta. Se pretende realizar la validación del modelo propuesto, utilizando el método Delphi, recibiendo ayuda de un grupo de expertos, los cuales serán seleccionados según el criterio de la autora de dicha investigación.

El método Delphi se abarca dentro de los métodos de prospectiva, utilizados frecuentemente como sistema para obtener información sobre el futuro(48).

Dentro de los métodos generales de prospectiva se destacan los: Métodos de expertos (método Delphi), métodos extrapolativos y métodos de correlación(49).

Métodos de expertos:

Los métodos de expertos posibilitan la estructuración de un proceso de comunicación grupal que es efectivo a la hora de permitir a un grupo de individuos, como un todo, tratar un problema complejo.

Método Delphi:

Se define como un método de estructuración de un proceso de comunicación grupal, efectivo a la hora de permitir a un grupo de individuos, como un todo, tratar un problema complejo. Una Delphi consiste en la selección de un grupo de expertos a los que se les pregunta su opinión sobre cuestiones referidas a acontecimientos del futuro. Las estimaciones de los expertos se realizan en sucesivas rondas, anónimas, al objeto de tratar de conseguir consenso, pero con la máxima autonomía por parte de los participantes(48).

La calidad de los resultados depende de la elaboración de los cuestionarios y la elección de los expertos consultados. El método Delphi procede por medio de la interrogación a expertos con la ayuda de cuestionarios sucesivos, a fin de poner de manifiesto convergencias de opiniones y deducir eventuales consensos(48).

Algunas de las ventajas que presenta el método Delphi son(49):

- Permite obtener información de puntos de vista sobre temas muy amplios o muy específicos. Los ejercicios Delphi son considerados “holísticos”, cubriendo una variedad muy amplia de campos.
- El horizonte de análisis puede ser variado.
- Permite la participación de un gran número de personas, sin que se forme el caos.
- Ayuda a explorar de forma sistemática y objetiva problemas que requieren la concurrencia y opinión cualificada.
- Elimina o aminora los efectos negativos de las reuniones de grupo “Cara-Cara”.

Previo a la aplicación del método se hace necesario delimitar el contexto y el horizonte temporal, en el que se desea realizar la previsión sobre el tema de estudio; seleccionar

el panel de expertos y conseguir su compromiso de colaboración además de explicar a los expertos en qué consiste el método(49).

Para la aplicación del método se siguieron tres etapas importantes:

1. Elección de expertos.
2. Elaboración del cuestionario, para la validación de la propuesta.
3. Análisis de resultados.

Elección de expertos:

Entiéndase por experto a la persona, con conocimientos relacionados con procesos de interoperabilidad, preparados para llegar a conclusiones objetivas y dar recomendaciones acerca de la presente investigación.

Los expertos fueron seleccionados según los siguientes criterios: Formación académica, debe ser graduado del nivel superior; Años de experiencia, debe tener un año de experiencia como mínimo; vinculación al producto SIAV; conocimientos básicos acerca de interoperabilidad entre aplicaciones, conocimiento acerca de estándares y patrones de comunicación, capacidad de análisis y pensamiento lógico. Estos criterios fueron seleccionados, a modo de asegurar que los expertos fueran personas preparadas, con al menos un previo conocimiento del tema de investigación. Esta selección brinda resultados con calidad, siempre que las opiniones acumuladas sean confiables y válidas para el objetivo propuesto.

Este método no cuenta con un valor exacto que determine el valor óptimo para el número de expertos que se debe seleccionar. Los investigadores de Rand Corporation expresan que es preciso contar con un mínimo de siete expertos o con un máximo de treinta. Con más de siete expertos el error disminuye exponencialmente; después de treinta el error disminuye de manera poco significativa y no compensa el incremento de costos y esfuerzo(21).

Para la selección de los expertos finales se debe hallar el grado de conocimiento de cada uno, con la ayuda del coeficiente de competencia.

Este coeficiente se determina mediante la fórmula: K= ½ (kc + ka), donde kc es el coeficiente de conocimientos y ka es el coeficiente de argumentación.

El coeficiente de conocimientos se obtiene de la siguiente tabla que recoge una

autoevaluación del posible experto.

							X			

0 1 2 3 4 5 6 7 8 9 10

Tabla 2. Tabla para calcular coeficiente de conocimiento.

El experto debe marcar según su criterio acerca de la capacidad que presenta sobre los procesos de interoperabilidad, en una escala del 0 al 10 y que después para ajustarla a la teoría de las probabilidades se multiplicará por 0,1.

Para calcular el coeficiente de argumentaciónel experto debe marcar en la siguiente tabla, según su criterio y su grado de competencia sobre los aspectos tratados.

FUENTES DE ARGUMENTACION	Grado de influencia de cada una de las fuentes en sus criterios.		
	A (alto)	M (medio)	B (bajo)
Análisis realizados por usted.		X	
Experiencia.	X		

Trabajos de autores nacionales.		X	
Trabajos de autores extranjeros.		X	
Su propio conocimiento del tema.			X
Su intuición.		X	

Tabla 3. Tabla para calcular coeficiente de argumentación.

Las marcas de los expertos se traducen a puntos, como se observa en la siguiente tabla:

FUENTES DE ARGUMENTACION	Grado de influencia de cada una de las fuentes en sus criterios.		
	A (alto)	M (medio)	B (bajo)
Análisis realizados por usted.	0.3	0.2	0.1

Experiencia.	0.5	0.4	0.2
Trabajos de autores nacionales.	0.05	0.05	0.05
Trabajos de autores extranjeros.	0.05	0.05	0.05
Su propio conocimiento del tema.	0.05	0.05	0.05
Su intuición.	1.0	0.8	0.5

Tabla 4. Tabla para calcular coeficiente de argumentación, llevada a puntos.

Donde este marcado con una X, se toma su valor en puntos, se suman todos y este es el valor de Ka.

El código para la interpretación de tales coeficientes de competencia es:

Si 0.8 < k <1.0, el coeficiente de competencia es alto.

Si 0.5 < k < 0.8, el coeficiente de competencia es medio.

Si k < 0.5 el coeficiente de competencia es bajo.

La forma descrita con anterioridad nos permite seleccionar la competencia de los expertos. Se lograron seleccionar de acuerdo a este análisis realizado un total de 8 expertos, donde tres estuvieron evaluados de medio y el resto de alto. En la siguiente figura se representa el resultado de acuerdo al coeficiente de competencia en general:

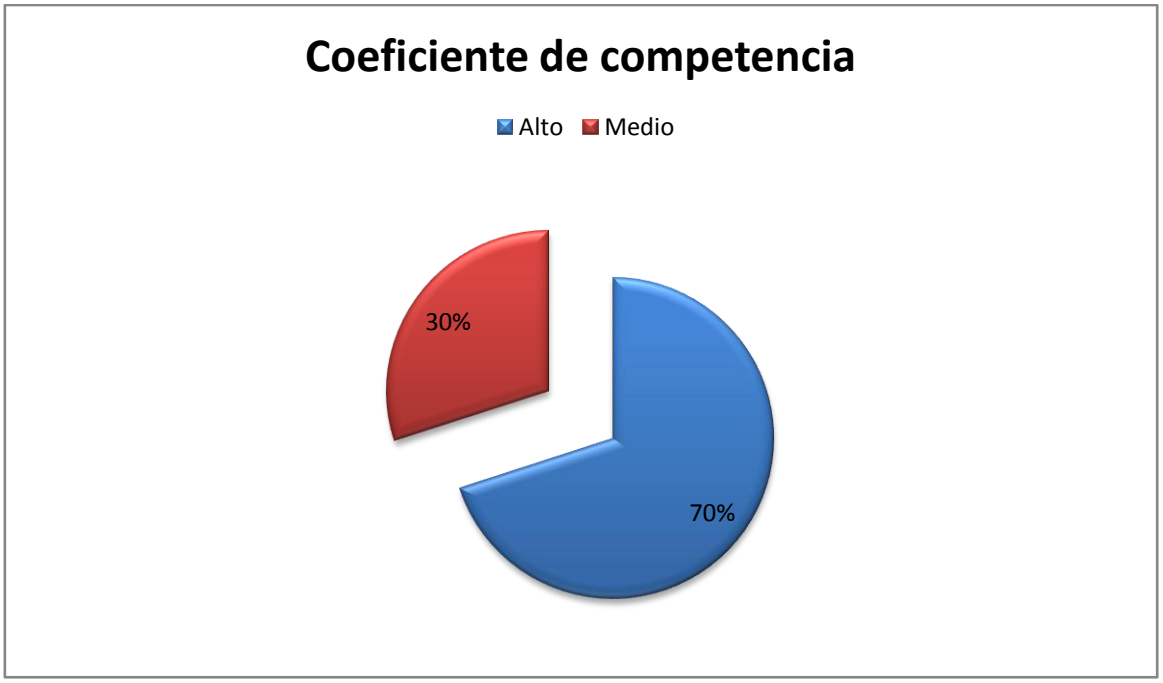


Figura 8. Coeficiente de competencia.

Logrado ya el número de expertos, se buscan sus criterios sobre la temática sometida a consideración. Las preguntas a formular no deben ser demasiadas, pero si sobre cuestiones referentes a la investigación que se realiza.

Se explica a cada experto, el modelo propuesto, para que sea capaz de entenderlo. Posteriormente lo debe evaluar mediante la encuesta realizada. Estos resultados se trataran finalmente en términos porcentuales.

Elaboración del cuestionario:

Los cuestionarios se elaboran de manera que faciliten la respuesta por parte de los encuestados.

Para la elaboración de las preguntas se tuvieron en cuenta tres objetivos generales:

- Demostrar que las tecnologías que se proponen como parte del funcionamiento del modelo, son suficientes para cumplir con los objetivos establecidos.
- Demostrar que el uso de un modelo de comunicación, aumenta potencialmente la comunicación; además aporta una solución para lograr el dinamismo necesario en el sistema.
- Demostrar que la propuesta constituye un aporte a la flexibilidad e interoperabilidad de los subsistemas de SIAV.

El cuestionario fue creado de forma tal que las respuestas fueran categorizadas en (muy adecuado (E1), bastante adecuado (E2), adecuado (E3), poco adecuado (E4) y no adecuado (E5), donde las An representan las afirmaciones, ver Anexo 1.

No.	Elementos	E1	E2	E3	E4	E5	Total
1	**A1**	5	2	1	0	0	8
2	**A2**	7	1	0	0	0	8
3	**A3**	3	3	2	0	0	8
4	**A4**	0	8	0	0	0	8
5	**A5**	6	2	0	0	0	8
6	**A6**	7	0	1	0	0	8
7	**A7**	6	1	1	0	0	8

Tabla 5. Tabla de frecuencias acumuladas.

La tabla de frecuencias acumuladas quedaría representada gráficamente como lo demuestra la siguiente figura.

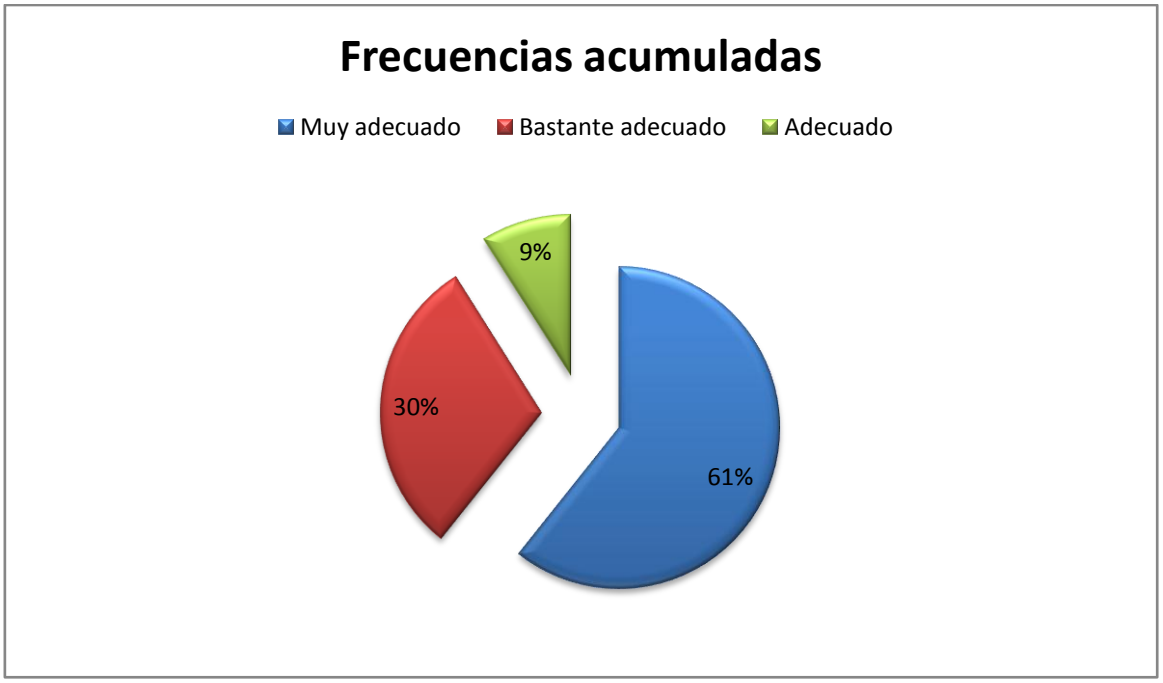


Figura 9. Representación gráfica de la tabla de frecuencias acumuladas.

Análisis de resultados:

Los resultados arrojados fueron satisfactorios, ya que un 61% de los expertos evaluaron la investigación de muy adecuada, 30% evaluaron de bastante adecuada y solo 9% de adecuada. De esta manera se aprecia que los expertos seleccionados están de acuerdo con el modelo propuesto.

Análisis de los resultados de la encuesta: La encuesta fue realizada con el objetivo de evaluar el dominio que posee el personal que trabaja en SIAV, acerca de las tecnologías y observar si era factible utilizar este modelo en SIAV. Realizarla permitió interactuar con el personal que más relacionado esta con el tema y darle credibilidad a la propuesta final. Después de analizar los resultados de la encuesta se reafirma la validez de la propuesta realizada. Los encuestados poseen un alto nivel de compresión y coinciden además en que el nivel de importancia concedido a las tecnologías es igual para todas. Además concuerdan en que hacer un modelo de comunicación dinámica, que use varias tecnologías es lo más adecuado para solucionar el problema de investigación.

3.3. Conclusiones Parciales

Sefundamentólaselecciónde lastecnologías a utilizar, para el desarrollo de un prototipo funcional,teniendoencuentalasnecesidadesexistentesenelSIAV. Se realizó un prototipo funcional que probo el funcionamiento del modelo de comunicación planteado. Se validó la propuesta de solución.Existe concordancia entre los expertos de que la aplicación de la propuesta realizada será un éxito e implicará numerosos beneficios.

Conclusiones

Una vez concluido el proceso investigativo, se puede garantizar que el desarrollo de un modelo de interoperabilidad, permite que los subsistemas de SIAV presenten una adecuada interacción.

Se obtuvo un modelo de comunicación que garantiza interacción entre los subsistemas de SIAV y a su vez con aplicaciones externas.

Se logró la especificación de un modelo de comunicación, que permite escalabilidad e interoperabilidad entre los componentes de software SIAV. El modelo especificado no está atado a una tecnología de interoperabilidad específica.

Se puede determinar que las tecnologías de interoperabilidad poseen ventajas y desventajas por lo que su selección estará fundamentada por el entorno de despliegue de los componentes de software desarrollados.

Se comprobó el funcionamiento del modelo de comunicación especificado. Los encuestados coinciden que es factible utilizar el modelo propuesto y que brinda una solución a las necesidades existentes en SIAV.

Se considera que el modelo contribuye con el aumento de la productividad y calidad en el proceso de desarrollo de software del departamento señales digitales.

Recomendaciones

En el estudio realizado sobre las técnicas actuales, con respecto a las tendencias existentes sobre la comunicación de aplicaciones, se puede observar la creciente importancia de lograr interoperabilidad en el desarrollo de soluciones informáticas. Para el perfeccionamiento y la continua evolución del modelo anteriormente planteado se recomienda lo siguiente:

Aplicar el modelo de comunicación en el desarrollo de componentes de software que comiencen a implementarse en el departamento de señales digitales.

Proponer el modelo de comunicación en las aplicaciones que se desarrollan en el centro GEySED en caso de considerarse factible.

Trabajos citados

1. **Santos, Luis Alberto.***Estándares de iure y de facto.*

2. **McKertich, Ashley.** eMarket services. [En línea] http://www.emarketservices.es/icex/cda/controller/pageemarket/0,3200,1480591_1515911_1517627_267664,00.html.

3. **Corporation, Microsoft.***La Arquitectura Orientada a Servicios (SOA) de Microsoft aplicada al mundo real.* 2006.

4. Definicion de. [En línea] 2008. http://definicion.de/aplicacion/.

5. **Szyperski, Clemens.***Component Software - Beyond Object-Oriented Programming.* 2002.

6. **Gomez, laureano Felipe.***Interoperabilidad en los sistemas de informacion documental.* Bogotá : s.n., 2007.

7. **Poggi, Eduardo.***Mas alla del E-government: La interoperabilidad y el Gobierno 2.0.* 2008.

8. **Puleo, Francisco.***Paradigmas de la información.* Mérida : s.n., 1985.

9. **Carballeria, Félix García.***Sistemas Distribuidos.* 1999.

10. Definición.de. *Definición de técnica.* [En línea] 2008. http://definicion.de/tecnica/.

11. **Bia, Alejandro.***APUNTES SOBRE PROCESOS Y DEADLOCK .* 1960.

12. **básica, Psicologia.***LENGUAJE Y COMUNICACIÓN.* 2007.

13. **colombia, Universidad nacional de.** Dirección nacional de servicios académicos virtuales. [En línea] 2012. http://www.virtual.unal.edu.co/cursos/sedes/manizales/4060010/lecciones/Capitulo1/modelo.htm.

14. **W3C.** w3c. [En línea] 2010. http://www.w3c.es/.

15. **Schwindt, Ariel.** Msdn microsoft. *Construcción de Sistemas Multiplataforma basados en Servicios .* [En línea] http://msdn.microsoft.com/es-es/library/bb972254.aspx.

16. **Esposito, Dino.** MSDN Magazine. *Creación de una capa de servicios AJAX segura .* [En línea] septiembre de 2008. http://msdn.microsoft.com/es-es/magazine/cc793961.aspx.

17. **Corporation, Microsoft.***Microsoft Application Architecture Guide, 2nd Edition.* 2009.

18. **Martínez, Evelio.** Eveliux.com. [En línea] julio de 21 de 2007.

http://www.eveliux.com/mx/protocolos-de-comunicaciones.php.

19. **BBS, Video Soft.** VSantivirus. *Falla en servicio RPC Endpoint Mapper.* [En línea] 27 de marzo de 2003. http://www.vsantivirus.com/vulms03-010.htm.

20. **Quevedo, Reneé Alejandro Gonzales y Cruz, Frank Najarro de la.***Definición de un modelo de integración de sistemas para la gestión de recursos en las empresas exportadoras aplicando SOA.* Lima : s.n., 2009.

21. **EcuRed.** EcuRed. *el RPC.* [En línea] http://www.ecured.cu/index.php/Llamada_a_Procedimiento_Remoto.

22. **Bernabé, Juan José Ortega.** Administración de sistemas informáticos en red. [En línea] 2011. http://dis.um.es/~lopezquesada/documentos/IES_1011/LMSGI/curso/material.html.

23. **Koftikian, Jack.***Simple Object Access Protocol.* Hamburg : s.n.

24. **Cerami, Ethan.***Web Services Essentials.* 2002.

25. **Gomez, Roberto.***XML-RPC.* 2009.

26. **Pañoso, Alberto Garcia.** Monitor de recursos multiplataformas con entorno de desarrollo. [En línea] 2005. http://DMReMon.com.

27. **Garrido, Hernan y Zunino, Cristian.***Seguridad en web services.* XML-RPC : s.n., 2006.

28. **Menza, Mario Alberto La.***Una introducción a Arquitectura Orientadas a Servicios (SOA) .*

29. **Muñoz, Jesús M. Conesa.***DESARROLLO DE UN SISTEMA PARA TOMA DE DECISIONES EN SITUACIONES DE INTRUSIÓN EN INSTALACIONES E INFRAESTRUCTURAS.* Madrid : s.n., 2009.

30. **Vallejo, David.***ZeroC ICE.* 2007.

31. **Manager, Free Download.**Free Download Manager. [En línea] 22 de septiembre de 2011. http://www.freedownloadmanager.org/es/downloads/Extensi%C3%B3n_de_lenguajes_comunes_para_Win32._80725_p/.

32. **Rosas, Javier Lunas.***Aplicando un algoritmo genetico para balancear carga dinamicamente en ambientes distribuidos orientados a objetos.* Es un middleware que permite a los clientes invocar operaciones de objetos distribuidos : s.n., 2003.

33. **Moncayo, Andrea.***Corba en Java.* Riombamba : s.n., 2011.

34. **Moya, Rodrigo, Robles, Gregorio y Majadas, Roberto.** Programación en el entorno GNOME. [En línea] 2002. http://www.calcifer.org/documentos/librognome/index.html.

35. **Marcos, Marga y Estevez, Elisabet.***Conceptos de Modelado.* Bilbao : s.n., 2010.

36. *Profundizando en estilos de arquitectura de software.* **Reynoso, Billy.** 2010, Architect academy.

37. **Clements, Paul.***Active reviews for intermediate designs.* 2000.

38. **Camacho, Erika y Cardeso, Fabio.***Arquitecturas de software.* 2004.

39. **Garlan, David.***An introduction to software architecture school of computer science.* 1994.

40. **Consortium, SCIPIO.** component-based development. [En línea] 1999. http://www.users.globalnet.co.uk/~rxv/CBDmain/cbdfaq.htm.

41. **Zelaya, Hazel Edith Ramirez y Reyes, Javier Alberto.***Arquitectura cliente/servidor.* 2010.

42. **Gracia, Joaquin.** Patrones de diseño. [En línea] 2005. http://www.ingenierosoftware.com/analisis y diseño.

43. **Welicki, León.** msdn.microsoft. [En línea] http://msdn.microsoft.com/es-es/library/bb972242.aspx.

44. **Campo, Gustavo Damian.***Patrones de Diseño, Refactorización y Antipatrones.* 2009.

45. **Gracia.** Patrones de diseño. . [En línea] 2005. http://www.ingenierosoftware.com/analisis y diseño.

46. 2mdc.com. [En línea] 2012. http://www.2mdc.com/.

47. Tedial. [En línea] 2011. http://www.tedial.com/.

48. **Astigarraga, Eneko.***EL MÉTODO DELPHI.* San Sebastian : s.n.

49. **Moreno, Cecilia Paula Izquierdo y Plaza, Beatriz Pascual.***El Método Delphi.*

50. **Henning, Michi.***Distributed Programming whit ICE.* 2008.

51. **Villate, Jaime E.***Introducción al XML.* Madrid : s.n., 2001.

52. *WSDL, el contrato de un servicio.* **Barco, Antonio.** diciembre de 2006.

53. **Collado, Cecilia.** Tecnologia escrita. *SO, primeros pasos.* [En línea] 2009. http://tecnologiaescrita.wordpress.com.

54. **Booch, Grady.***el lenguaje unificado de modelado.* 2003.

55. **Jalon, Javier Garcia de, Rodriguez, Jose Ignacio y Goñi, Rufino.***aprenda c++ como si estuviera en primero.* San Sebastian : s.n., 1998.

56. **Stallings, William.***Sistemas operativos: aspectos internos y principios de diseño.* 2007.

57. **Marvin, David.***Definicion del lenguaje de programación.* 2008.

58. **Giraldo, Luis.***Herramientas de desarrollo de ingenieria de software para Linux.*

2005.

59. **Pressman, Roger.***Ingenieria de Software:un enfoque practico.* 2002.

60. **Harbou, Michael González.***middleware de distribucion y modelo transaccional en sistemas de tiempo real.* 2005.

61. **Roca, Vicent.***RTP/RTCP and RTSP multimedia protocols for the Internet.* 2001.

62. **Gamma, Erich.***Desing patterns.* 1995.

63. Documentacion arquitectonica de Microsoft: Patterns and Practices. [En línea] http://www.microsoft.com/spanish/msdn/arquitectura.

64. **Buschmann, Frank.***Patterns oriented software architecture.* 1996.

65. **Bass, Len.***Software architecture in practice.* 2003.

66. **Shaw, Mary.***Software architecture.* 1999.

67. **Thilmany, Christian.***.NET patterns: architecture, desing and process.* 2003.

68. **Orallo.***El lenguaje unificado de modelado (UML).* 2002.

69. **Alonso.***Web Services Concepts, Architectures and Applications.* 2004.

70. **Meyer, Lisandro Damián Nicanor Pérez.***Introducción al desarrollo multiplataforma con QT.* 2007.

71. **Corporation, Nokia.** QT . [En línea] 2008-2012. http://qt.nokia.com/products/developers-tools.

72. **Pelaez, Juan.** Arquitectura basada en Componentes. [En línea] 2009. http://www.juanpelaez.com/geek-stuff/arquitectura/arquitectura-basada-en-componentes/.

73. **Sosa, Víctor J. Sosa.***MIDDLEWARE: Arquitectura para Aplicaciones Distribuidas.* 2006.

74. **Ruggia, Raul.***Interoperabilidad entre Servidores de Aplicaciones Heterogéneos.* Montevideo : s.n., 2008.

75. **Olarte, Carlos A.***CORBA, Una arquitectura para integrar ambientes distribuidos y heterog´eneos.* Madrid : s.n., 2010.

76. **Sams.***Sams Teach Yourself CORBA in 14 Days.* 1998.

77. **Shklar, Leon.***Web Application Architecture.* England : John Wiley & Sons Ltd,, 2003.

78. **CALI, A. D.** Glosario A-C de la Infraestructura de Datos Espaciales de Santiago de Cali. [En línea] 2011. http://www.cali.gov.co/planeacion/publicaciones.php?id=33568.

79. **EcuRed.** Conocimiento con todos y para todos. [En línea] 2012. http://www.ecured.cu/index.php/Qt.

80. **Larman, Craig.***UML y Patrones. Introducción al análisis y diseño orientado a*

objetos. Primera edición. Prentice Hall, México : s.n., 1999.

81. ArcGIS 9.2 Desktop Help. About selecting features by locations. [En línea] Enero de 2008.

82. **Pressman, Roger S.***Ingeniería de Software un Enfoque Práctico.* 2000. Vol. 6ta Edición.

83. **Arias, Fidias G.***EL PROYECTO DE INVESTIGACION.* caracas : Editorial Episteme, 1999.

84. **Surroca, Alfredo.***Arquitectura orientada a servicios.* Madrid : Accenture, 2008.

85. **corporation, ArcGIS.** ArcGIS Resource Center. [En línea] 2011. http://help.arcgis.com/es/arcgisdesktop/10.0/help/index.htm.

86. **Piñol, Carles Mateu.***Desarrollo de aplicaciones web.*

87. **Ariel Schwindt.** msdn.microsoft. *Construcción de Sistemas Multiplataforma basados en Servicios .* [En línea] http://msdn.microsoft.com/es-es/library/bb972254.aspx.

88. **Sun, Xian-He y Blatecky, Alan R.***Preface: middleware: the key to next generation computing.* Orlando, FL : s.n., 2004.

89. **shklar, Leon y rosen, Richard.***Web application architecture.* 2003.

90. **Berners-Lee, T.***Hypertext Transfer Protocol -- HTTP/1.1.* 1999.

91. **Gutiérrez., Javier J.***¿Qué es un framework web?*

92. **Alexander, Christopher.***The Timeless Way of Building.* 1979.

Bibliografía

1. **Santos, Luis Alberto.***Estándares de iure y de facto.*

2. **McKertich, Ashley.** eMarket services. [En línea] http://www.emarketservices.es/icex/cda/controller/pageemarket/0,3200,1480591_1515911_1517627_267664,00.html.

3. **Corporation, Microsoft.***La Arquitectura Orientada a Servicios (SOA) de Microsoft aplicada al mundo real.* 2006.

4. Definicion de. [En línea] 2008. http://definicion.de/aplicacion/.

5. **Szyperski, Clemens.***Component Software - Beyond Object-Oriented Programming.* 2002.

6. **Gomez, laureano Felipe.***Interoperabilidad en los sistemas de informacion documental.* Bogotá : s.n., 2007.

7. **Poggi, Eduardo.***Mas alla del E-government: La interoperabilidad y el Gobierno 2.0.* 2008.

8. **Puleo, Francisco.***Paradigmas de la información.* Mérida : s.n., 1985.

9. **Carballeria, Félix García.***Sistemas Distribuidos.* 1999.

10. Definición.de. *Definición de técnica.* [En línea] 2008. http://definicion.de/tecnica/.

11. **Bia, Alejandro.***APUNTES SOBRE PROCESOS Y DEADLOCK .* 1960.

12. **básica, Psicologia.***LENGUAJE Y COMUNICACIÓN.* 2007.

13. **colombia, Universidad nacional de.** Dirección nacional de servicios académicos virtuales. [En línea] 2012. http://www.virtual.unal.edu.co/cursos/sedes/manizales/4060010/lecciones/Capitulo1/modelo.htm.

14. **W3C.** w3c. [En línea] 2010. http://www.w3c.es/.

15. **Schwindt, Ariel.** Msdn microsoft. *Construcción de Sistemas Multiplataforma basados en Servicios .* [En línea] http://msdn.microsoft.com/es-es/library/bb972254.aspx.

16. **Esposito, Dino.** MSDN Magazine. *Creación de una capa de servicios AJAX segura .* [En línea] septiembre de 2008. http://msdn.microsoft.com/es-es/magazine/cc793961.aspx.

17. **Corporation, Microsoft.***Microsoft Application Architecture Guide, 2nd Edition.* 2009.

18. **Martínez, Evelio.** Eveliux.com. [En línea] julio de 21 de 2007.

http://www.eveliux.com/mx/protocolos-de-comunicaciones.php.

19. **BBS, Video Soft.** VSantivirus. *Falla en servicio RPC Endpoint Mapper.* [En línea] 27 de marzo de 2003. http://www.vsantivirus.com/vulms03-010.htm.

20. **Quevedo, Reneé Alejandro Gonzales y Cruz, Frank Najarro de la.***Definición de un modelo de integración de sistemas para la gestión de recursos en las empresas exportadoras aplicando SOA.* Lima : s.n., 2009.

21. **EcuRed.** EcuRed. *el RPC.* [En línea] http://www.ecured.cu/index.php/Llamada_a_Procedimiento_Remoto.

22. **Bernabé, Juan José Ortega.** Administración de sistemas informáticos en red. [En línea] 2011. http://dis.um.es/~lopezquesada/documentos/IES_1011/LMSGI/curso/material.html.

23. **Koftikian, Jack.***Simple Object Access Protocol.* Hamburg : s.n.

24. **Cerami, Ethan.***Web Services Essentials.* 2002.

25. **Gomez, Roberto.***XML-RPC.* 2009.

26. **Pañoso, Alberto Garcia.** Monitor de recursos multiplataformas con entorno de desarrollo. [En línea] 2005. http://DMReMon.com.

27. **Garrido, Hernan y Zunino, Cristian.***Seguridad en web services.* XML-RPC : s.n., 2006.

28. **Menza, Mario Alberto La.***Una introducción a Arquitectura Orientadas a Servicios (SOA) .*

29. **Muñoz, Jesús M. Conesa.***DESARROLLO DE UN SISTEMA PARA TOMA DE DECISIONES EN SITUACIONES DE INTRUSIÓN EN INSTALACIONES E INFRAESTRUCTURAS.* Madrid : s.n., 2009.

30. **Vallejo, David.***ZeroC ICE.* 2007.

31. **Manager, Free Download.**Free Download Manager. [En línea] 22 de septiembre de 2011. http://www.freedownloadmanager.org/es/downloads/Extensi%C3%B3n_de_lenguajes_comunes_para_Win32._80725_p/.

32. **Rosas, Javier Lunas.***Aplicando un algoritmo genetico para balancear carga dinamicamente en ambientes distribuidos orientados a objetos.* Es un middleware que permite a los clientes invocar operaciones de objetos distribuidos : s.n., 2003.

33. **Moncayo, Andrea.***Corba en Java.* Riombamba : s.n., 2011.

34. **Moya, Rodrigo, Robles, Gregorio y Majadas, Roberto.** Programación en el entorno GNOME. [En línea] 2002. http://www.calcifer.org/documentos/librognome/index.html.

35. **Marcos, Marga y Estevez, Elisabet.***Conceptos de Modelado.* Bilbao : s.n., 2010.

36. *Profundizando en estilos de arquitectura de software.* **Reynoso, Billy.** 2010, Architect academy.

37. **Clements, Paul.***Active reviews for intermediate designs.* 2000.

38. **Camacho, Erika y Cardeso, Fabio.***Arquitecturas de software.* 2004.

39. **Garlan, David.***An introduction to software architecture school of computer science.* 1994.

40. **Consortium, SCIPIO.** component-based development. [En línea] 1999. http://www.users.globalnet.co.uk/~rxv/CBDmain/cbdfaq.htm.

41. **Zelaya, Hazel Edith Ramirez y Reyes, Javier Alberto.***Arquitectura cliente/servidor.* 2010.

42. **Gracia, Joaquin.** Patrones de diseño. [En línea] 2005. http://www.ingenierosoftware.com/analisis y diseño.

43. **Welicki, León.** msdn.microsoft. [En línea] http://msdn.microsoft.com/es-es/library/bb972242.aspx.

44. **Campo, Gustavo Damian.***Patrones de Diseño, Refactorización y Antipatrones.* 2009.

45. **Gracia.** Patrones de diseño. . [En línea] 2005. http://www.ingenierosoftware.com/analisis y diseño.

46. 2mdc.com. [En línea] 2012. http://www.2mdc.com/.

47. Tedial. [En línea] 2011. http://www.tedial.com/.

48. **Astigarraga, Eneko.***EL MÉTODO DELPHI.* San Sebastian : s.n.

49. **Moreno, Cecilia Paula Izquierdo y Plaza, Beatriz Pascual.***El Método Delphi.*

50. **Henning, Michi.***Distributed Programming whit ICE.* 2008.

51. **Villate, Jaime E.***Introducción al XML.* Madrid : s.n., 2001.

52. *WSDL, el contrato de un servicio.* **Barco, Antonio.** diciembre de 2006.

53. **Collado, Cecilia.** Tecnologia escrita. *SO, primeros pasos.* [En línea] 2009. http://tecnologiaescrita.wordpress.com.

54. **Booch, Grady.***el lenguaje unificado de modelado.* 2003.

55. **Jalon, Javier Garcia de, Rodriguez, Jose Ignacio y Goñi, Rufino.***aprenda c++ como si estuviera en primero.* San Sebastian : s.n., 1998.

56. **Stallings, William.***Sistemas operativos: aspectos internos y principios de diseño.* 2007.

57. **Marvin, David.***Definicion del lenguaje de programación.* 2008.

58. **Giraldo, Luis.***Herramientas de desarrollo de ingenieria de software para Linux.*

2005.

59. **Pressman, Roger.***Ingenieria de Software:un enfoque practico.* 2002.

60. **Harbou, Michael González.***middleware de distribucion y modelo transaccional en sistemas de tiempo real.* 2005.

61. **Roca, Vicent.***RTP/RTCP and RTSP multimedia protocols for the Internet.* 2001.

62. **Gamma, Erich.***Desing patterns.* 1995.

63. Documentacion arquitectonica de Microsoft: Patterns and Practices. [En línea] http://www.microsoft.com/spanish/msdn/arquitectura.

64. **Buschmann, Frank.***Patterns oriented software architecture.* 1996.

65. **Bass, Len.***Software architecture in practice.* 2003.

66. **Shaw, Mary.***Software architecture.* 1999.

67. **Thilmany, Christian.***.NET patterns: architecture, desing and process.* 2003.

68. **Orallo.***El lenguaje unificado de modelado (UML).* 2002.

69. **Alonso.***Web Services Concepts, Architectures and Applications.* 2004.

70. **Meyer, Lisandro Damián Nicanor Pérez.***Introducción al desarrollo multiplataforma con QT.* 2007.

71. **Corporation, Nokia.** QT . [En línea] 2008-2012. http://qt.nokia.com/products/developers-tools.

72. **Pelaez, Juan.** Arquitectura basada en Componentes. [En línea] 2009. http://www.juanpelaez.com/geek-stuff/arquitectura/arquitectura-basada-en-componentes/.

73. **Sosa, Víctor J. Sosa.***MIDDLEWARE: Arquitectura para Aplicaciones Distribuidas.* 2006.

74. **Ruggia, Raul.***Interoperabilidad entre Servidores de Aplicaciones Heterogéneos.* Montevideo : s.n., 2008.

75. **Olarte, Carlos A.***CORBA, Una arquitectura para integrar ambientes distribuidos y heterog´eneos.* Madrid : s.n., 2010.

76. **Sams.***Sams Teach Yourself CORBA in 14 Days.* 1998.

77. **Shklar, Leon.***Web Application Architecture.* England : John Wiley & Sons Ltd,, 2003.

78. **CALI, A. D.** Glosario A-C de la Infraestructura de Datos Espaciales de Santiago de Cali. [En línea] 2011. http://www.cali.gov.co/planeacion/publicaciones.php?id=33568.

79. **EcuRed.** Conocimiento con todos y para todos. [En línea] 2012. http://www.ecured.cu/index.php/Qt.

80. **Larman, Craig.***UML y Patrones. Introducción al análisis y diseño orientado a*

objetos. Primera edición. Prentice Hall, México : s.n., 1999.

81. ArcGIS 9.2 Desktop Help. About selecting features by locations. [En línea] Enero de 2008.

82. **Pressman, Roger S.***Ingeniería de Software un Enfoque Práctico.* 2000. Vol. 6ta Edición.

83. **Arias, Fidias G.***EL PROYECTO DE INVESTIGACION.* caracas : Editorial Episteme, 1999.

84. **Surroca, Alfredo.***Arquitectura orientada a servicios.* Madrid : Accenture, 2008.

85. **corporation, ArcGIS.** ArcGIS Resource Center. [En línea] 2011. http://help.arcgis.com/es/arcgisdesktop/10.0/help/index.htm.

86. **Piñol, Carles Mateu.***Desarrollo de aplicaciones web.*

87. **Ariel Schwindt.** msdn.microsoft. *Construcción de Sistemas Multiplataforma basados en Servicios .* [En línea] http://msdn.microsoft.com/es-es/library/bb972254.aspx.

88. **Sun, Xian-He y Blatecky, Alan R.***Preface: middleware: the key to next generation computing.* Orlando, FL : s.n., 2004.

89. **shklar, Leon y rosen, Richard.***Web application architecture.* 2003.

90. **Berners-Lee, T.***Hypertext Transfer Protocol -- HTTP/1.1.* 1999.

91. **Gutiérrez., Javier J.***¿Qué es un framework web?*

92. **Alexander, Christopher.***The Timeless Way of Building.* 1979.

Anexos

Anexo 1

Encuesta Fecha:20/05/2012

Realizadapor:

NombreyApellidos	Cargo
Ivón Martínez Genis	Estudiante de quinto año

Encuestados:

NombreyApellidos	Cargo
Abel Díaz Berenguer	LíderdeProyecto
Dayami Chávez Ayala	Analista del producto SIAV
Adnan Díaz Fuentes	Desarrollador
Jean Michael Suarez Pérez	Desarrollador
Yoandris Silverio Pacheco Jerez	Desarrollador
Frank Torres Rodríguez	Desarrollador
Eduardo Cepero Utra	Desarrollador
Ekaterina Alejovna Ramírez Ramírez	Desarrollador

Objetivo:

Lapresenteencuestaserealizaconelobjetivodeconocerque creen los profesores de SIAV, acerca del modelo propuesto. Se propone dar a conocer si el modelo propuesto tendrá éxitos, mediante la evaluación dada por cada profesor a las afirmaciones planteadas en el cuestionario.

A continuación se relacionan un conjunto de afirmaciones, valore su factibilidad, marcando con una X de acuerdo a:

E1: Muy adecuado

E2: Bastante adecuado

E3: Adecuado

E4: Poco adecuado

E5: No adecuado

Pregunta1: Considera que la realización de la propuesta de un modelo de

comunicación, para aplicar en SIAV es:

Pregunta 2: Evalúe el desarrollo de las tecnologías mediante plug-ins en:

Pregunta 3: Usando las tecnologías XML-RPC, ICE y SOAP, se valida la propuesta de manera:

Pregunta 4: Con la implementación del modelo de comunicación, se logra una interoperabilidad entre los subsistemas de SIAV:

Pregunta 5: La aplicación del modelo en todo el departamento, tendrá posibilidades:

Pregunta 6: La utilización del modelo propuesto, en los proyectos de Señales digitales, tiene importancia:

Pregunta 7: La propuesta está a la altura de las necesidades y posibilidades de SIAV de manera:

		E1	E2	E3	E4	E5
A1	Pregunta 1					
A2	Pregunta 2					
A3	Pregunta 3					
A4	Pregunta 4					
A5	Pregunta 5					
A6	Pregunta 6					
A7	Pregunta 7					