UNIVERSIDAD CENTRAL "MARTA ABREU" DE LAS VILLAS
FACULTAD DE MATEMÁTICA, FÍSICA Y COMPUTACIÓN
CENTRO DE ESTUDIOS DE INFORMÁTICA

``Arquitectura para la integración de aplicaciones de la Facultad de Matemática, Física y Computación y el Centro de Estudios de Informática´´

Tesis presentada en opción al título de Licenciado en Ciencia de la Computación

Autores:
Sady Pérez Cabrera
Claudia Fernández Hernández

Tutores:
Dr. Carlos García González
M.Sc. Eladio Cuellar

Santa Clara, Cuba, 2014

DEDICATORIA

A mis padres Miriam Cabrera y Orlando Pérez que han sido unos padres excelentes, que me han demostrado a ir por la vida sin tener miedo a los obstáculos y que nunca les ha faltado un momento de comprensión y amor.

Sady.

A mi familia, por ser tan importante en mi vida y estar presente en todo momento.

Claudia.

AGRADECIMIENTOS

Quisiéramos agradecer a todas aquellas personas que han contribuido de una forma u otra en el desarrollo de este trabajo de diploma, particularmente a nuestro tutor Dr. Carlos García, a Miriam y Manuel Nicado, y Orestes Febles, quién sin conocernos nos brindó su ayuda.

Individualmente, quisiéramos también agradecer a varias personas.

Quiero agradecer a mis padres Miriam y Orlando, que supieron apoyarme ante cualquier decisión, a mi familia pues todos me han ayudado, a mi novio Ernesto que siempre estuvo para animarme a seguir adelante, a Mirian y Luis que desde siempre han estado incondicional para mí. A Yanet y Yoa que me guiaron durante estos 5 años. Por supuesto, a mis amigos: Salvador, Luis Javier, Víctor, Juan Luis, Luis Miguel, Dueña, José Manuel, Jennifer (Lucy), Daynel y Luis Felipe, porque sin ellos la universidad nunca hubiese sido la misma, son insustituibles para mí y han demostrado ser verdaderos amigos.

Sady.

A mi familia, por estar a mi lado siempre que lo he necesitado, brindándome apoyo y seguridad, en especial a mis padres Ileana y Feliberto. A mi novio Manuel, por su preocupación constante. A Liliana y a Juan Manuel, por su amistad. A Jennifer, Osnam (Noni) y Juan Luis, otros amigos que también me han ayudado.

Claudia.

RESUMEN

Actualmente en la facultad Matemática, Física y Computación (MFC) y en el Centro de Estudios de Informática (CEI) de la Universidad Central "Marta Abreu" de Las Villas se mantienen varios sistemas de información que apoyan el trabajo que se realiza en ambos centros. El estudio del estado actual de la informatización en el CEI y la facultad MFC permitió identificar problemas de duplicación de datos y funcionalidad, así como ausencia de interoperabilidad, resultados de un desarrollo aislado e independiente de los sistemas. Con el fin de lograr la integración entre los sistemas se propone la aplicación de una Arquitectura Orientada a Servicios (por sus siglas en inglés SOA). El desarrollo de la solución SOA abarca las etapas de análisis y diseño, lo que permitirá la identificación y descripción de los servicios, componentes principales de la arquitectura. Se expone el diseño de la infraestructura que sostendrá la implementación y utilización de los servicios. En el caso particular del sitio web del CEI, la necesidad de acceder a la información almacenada en el repositorio DSpace para la implementación de una de sus funcionalidades, presenta un nuevo escenario de integración. Para lograr la interoperabilidad entre estos sistemas se propone el empleo del protocolo de intercambio de información OAI-PMH. Basado en el modelo de recolección de metadatos, este protocolo permite la recuperación de los datos desde el repositorio DSpace para su posterior exposición en el sitio web del CEI.

ABSTRACT

Currently, in the Mathematics, Physics and Computer Science faculty (MFC) and in the Center for Computer Studies is maintained a number of information systems that support the work of both centers. The study of the current state of computerization in the CEI and the MFC faculty allowed the identification of problems of duplication of data and functionality as well as the lack of interoperability, all of them results of an isolated and independent development of the systems. In order to achieve the integration among systems, the application of a Service-Oriented Architecture (SOA) is proposed. The development of the SOA solution covers the analysis and design stages, allowing the identification and description of services, main components of the SOA. The design of the infrastructure that will support the implementation and use of services is proposed. In the particular case of the CEI's website, the need to access the information stored in the DSpace repository to implement one of its features introduces a new integration scenario. To achieve interoperability among these systems is proposed the use of the information exchange protocol OAI-PMH. Based on the metadata harvesting model, this protocol enables the recovery of data from the DSpace repository for subsequent display on the CEI's website.

TABLA DE CONTENIDOS

TABLA DE FIGURAS

INTRODUCCIÓN

En la actualidad para las instituciones la información y los contenidos digitales son cada vez más importantes. Cuando las instituciones son tan dependientes de la información, es muy importante que la tecnología que proporciona esta información y que se emplea para apoyar los procesos que se desarrollan en las mismas, esté en consonancia con las necesidades de la propia institución. Otro problema que ha ido alcanzado gran relevancia dentro de las instituciones, dado la creciente dependencia a la información, es la duplicación de datos y funcionalidades, resultado, principalmente, de una estructuración aislada e independiente de las tecnologías de la información (TI): sistemas, aplicaciones y datos.

En este marco, la integración de aplicaciones ha incrementado su importancia debido a que frecuentemente las TI en las instituciones adoptan la forma de islas de información, lo que ocasiona que el valor de los sistemas individuales no sea aprovechado al máximo debido a su aislamiento. La falta de integración entre los componentes de las TI hace difícil obtener una respuesta rápida y efectiva ante los cambios que afectan de forma natural a los negocios. La inflexibilidad genera costos, reduce la capacidad de respuesta ante los clientes, compromete el cumplimiento con las normativas legales y afecta negativamente la productividad de los empleados. En resumen, una deficiente integración es uno de los problemas más importantes a los que las organizaciones deben hacer frente para mantener su competitividad y garantizar su crecimiento(Microsoft, 2006).

Si bien la integración es una necesidad para las instituciones en la actualidad, de aplicarse sin seguir un enfoque estructurado, las conexiones punto a punto crecerían al interior de la organización resultando en una masa irregular que es difícil de mantener. De no tomarse las decisiones correctas pudiera conducir a problemas en el rendimiento, el funcionamiento y la calidad de los servicios.

Actualmente en la Facultad de Matemática, Física y Computación (MFC) y el Centro de Estudios de Informática (CEI) se encuentran en explotación varias aplicaciones, entre las que pueden citar: la intranet de la Facultad (http://fmfc.uclv.edu.cu), la intranet del CEI (http://ceinf.uclv.edu.cu) con sitios para todos los grupos de investigación, el Sistema de Control de Medios Básicos para el CEI, el Sistema de control de Horario Docentes, el Sistema de

Información de apoyo a la Autoevaluación de la Carrera Ciencia de la Computación y el Sistema para el Control de la Carga Docente en el Departamento de Ciencia de la Computación.

La mayoría de estas aplicaciones han sido desarrolladas de manera aislada y utilizando en ocasiones plataformas de desarrollo diferentes, y como consecuencia se identifican los siguientes problemas:

- Los datos se encuentran dispersos por diferentes sitios con gran redundancia e inconsistencia entre ellos.
- Al desarrollarse las aplicaciones como islas, no existe reutilización de funcionalidades que son comunes.
- Como las tecnologías utilizadas muchas veces varían entre una aplicación y otra, se hace muy difícil el entendimiento entre las mismas.
- Al existir gran redundancia el esfuerzo de mantenimiento de la información se multiplica.

Por otra parte, desde hace varios años, los directivos del CEI utilizan un repositorio digital institucional para almacenar la información que se genera como parte del propio trabajo del centro. Esto incluye información de tipo administrativa, académica y de ciencia y técnica. En la implementación del repositorio se utilizó el software de código abierto DSpace.

Dentro de los servicios que debe exponer el sitio del CEI está la conformación del currículo vitae de los profesores e investigadores, el cual debe incluir también la producción científica de los mismos. Esta facilidad aún no ha sido implementada pues para conformar la producción científica de un profesor o investigador es necesario acceder a la información contenida en el repositorio del DSpace. Esta acción no resulta evidente ya que una parte de la información del repositorio se almacena en una base de datos relacional, empleando como sistema gestor PostgreSQL, y el resto en carpetas sobre el sistema de archivos del sistema operativo (Windows o Linux). En este caso se identifica un problema de interoperabilidad entre el sitio del CEI y el repositorio del DSpace.

La existencia de redundancia a nivel de los datos entre las diversas aplicaciones que actualmente se despliegan en el CEI y la facultad de MFC, y la falta de interoperabilidad entre algunos de estos sistemas, constituye el ***problema de investigación*** de esta tesis que tiene como ***objetivo general***:

Diseñar una arquitectura que integre las aplicaciones a nivel de la Facultad de MFC y del CEI identificando las tecnologías que mejor se adapten a las problemáticas identificadas.

Para su cumplimiento, el objetivo general se desglosa en los siguientes ***objetivos específicos***:

1. Realizar un estudio del estado actual de la informatización en la facultad de MFC y el CEI.
2. Identificar y valorar las principales arquitecturas de integración que se pueden aplicar a las problemáticas identificada.
3. Diseñar una arquitectura de integración que elimine los problemas existentes.
4. Implementar, una alternativa de acceso a la información contenida en el repositorio del DSpace desde el sitio web del CEI.

Las **preguntas de investigación** planteadas están relacionadas sobre:

1. ¿Cómo concebir una arquitectura de integración que elimine la redundancia e inconsistencia entre los datos?
2. ¿Qué servicios deben ser implementados para permitir el intercambio de información entre las diversas aplicaciones?
3. ¿Qué tecnologías están disponibles para el acceso desde otras aplicaciones a la información contenida en el repositorio que gestiona el sistema DSpace?

Justificación de la investigación

La utilización de una arquitectura que integre y permita la reutilización de funcionalidades entre las diversas aplicaciones de la facultad de MFC y del CEI, contribuye a eliminar los problemas que se derivan de una estructura aislada, con poca o ninguna interoperabilidad entre sus aplicaciones y posibilita la inserción de nuevas aplicaciones en el marco de esta arquitectura, con lo cual se justifica el desarrollo de esta investigación.

La relevancia social de este proyecto radica precisamente en la contribución a la propuesta de una arquitectura de integración que elimine los problemas reales existentes en la facultad de MFC y del CEI y posibilite mejoras a la estructura actual. Adicionalmente, la divulgación en el sitio web del CEI de la producción científica de sus profesores e investigadores es un aspecto muy importante que posibilita una mayor visibilidad y disponibilidad científica, pues permite dar a conocer a la comunidad internacional los resultados científicos del CEI. Por otra parte, las

implicaciones prácticas se basan en el estudio y la aplicación de patrones de integración, así como de diversos estándares para la solución de problemáticas reales.

La **tesis** está **estructurada** en tres capítulos. En el primer capítulo se presentan los fundamentos teóricos del trabajo, estudio y actualidad de arquitecturas y principios para la integración de aplicación. Se abordan algunos protocolos para el intercambio de información como una manera de permitir la interoperabilidad entre aplicaciones. Se emiten valoraciones y criterios que definen la propuesta de solución. En el segundo capítulo se detallan los procesos de análisis y diseño para la arquitectura de integración propuesta. En el tercer capítulo se presenta la solución para el caso particular del repositorio del DSpace y el sitio web del CEI que permite la interoperabilidad entre ambas aplicaciones. Por último, se formulan las conclusiones y recomendaciones.

CAPÍTULO 1. Marco teórico referencial.

En el capítulo se presenta, primeramente, un análisis del estado actual de la informatización en el Centro de Estudios de Informática y la facultad de Matemática, Física y Computación. Se examinan los principales estilos o estrategias adoptados como parte del progresivo desarrollo de las iniciativas de integración de aplicaciones. Se analizan las características de varios importantes estilos de integración, destacando las ventajas y desventajas de su aplicación en el desarrollo de aplicaciones integradas. Se detalla el empleo de los servicios web como una alternativa de implementación de la arquitectura orientada a servicios. Se aborda el tema de los protocolos de intercambio de información, como principal mecanismo para el intercambio de datos e información entre aplicaciones o sistemas.

1.1 Estado de la informatización del CEI y la facultad de MFC

Como es tradicional dentro de las instituciones, la Facultad de Matemática, Física y Computación (MFC) y el Centro de Estudios de Informática (CEI) se encuentran estructurados funcionalmente. Esto significa que existen varios departamentos para desarrollar diferentes funciones. Todos estos departamentos utilizan sus propios sistemas de TI que hacen un seguimiento de los datos que se necesitan como parte del propio trabajo de ambos centros.

Las principales aplicaciones que se encuentran en explotación y que se precisan como apoyo en la labor docente y administrativa tanto del CEI, como de la facultad de MFC son:

- Sistema de Control de Medios Básicos para el CEI
- Sistema de Control de Horario Docentes para la facultad de MFC
- Sistema de Información de Apoyo a la Autoevaluación de la Carrera Ciencia de la Computación
- Sistema para el Control de la Carga Docente en el Departamento de Ciencia de la Computación.
- Intranet del CEI (http://ceinf.uclv.edu.cu)
- Intranet de la Facultad (http://fmfc.uclv.edu.cu)
- Repositorio DSpace

Seguidamente se presentarán las principales característica de organización y funcionamiento de estos sistemas, lo que permitirá y sostendrá la realización de un diagnóstico del estado actual de la informatización.

1.1.1 Sistema de Control de Medios Básicos para el CEI.

El Sistema de Control de Medios Básicos facilita el manejo de información administrativa de CEI, al mantener los datos vinculados con la utilización y el estado de los medios básicos con que cuenta el centro.

La figura 1.1 muestra el modelo físico de la base de datos que presenta información que maneja el sistema.

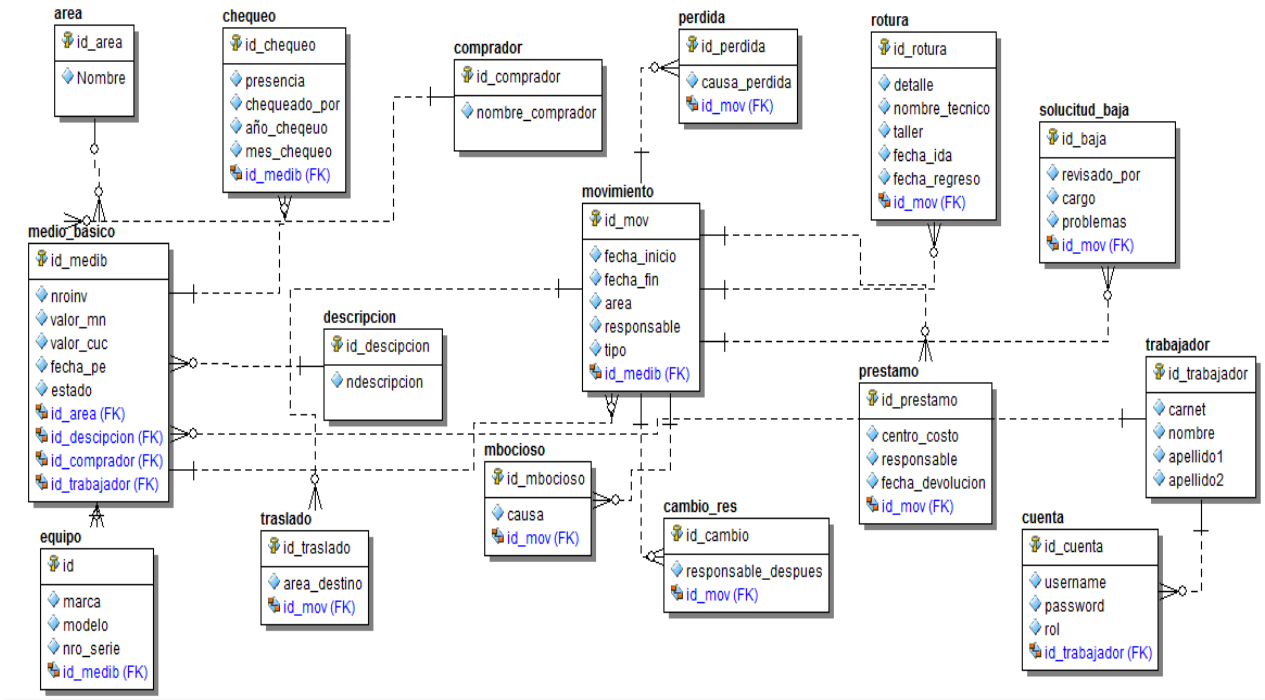


Figura 1.1. Modelo Físico para el Sistema de Control de Medios Básicos

1.1.2 Sistema de control de Horarios Docentes.

El Sistema de Control de Horarios Docentes satisface la necesidad de publicar de forma automática y de libre acceso por estudiantes, profesores y directivos la planificación del horario docente, tanto para el uso de la infraestructura de aulas y laboratorios como para el correcto control de la planificación según los planes de estudios de las distintas carreras de la facultad de MFC (González, 2013).

La publicación del horario docente teniendo en cuenta todas las restricciones de asignaturas, años, profesores, hora de clases, etc. facilita el trabajo del personal relacionado con el proceso docente - educativo.

La figura 1.2 muestra el diagrama Entidad-Relación que permite exponer la información que maneja el sistema (González, 2013).

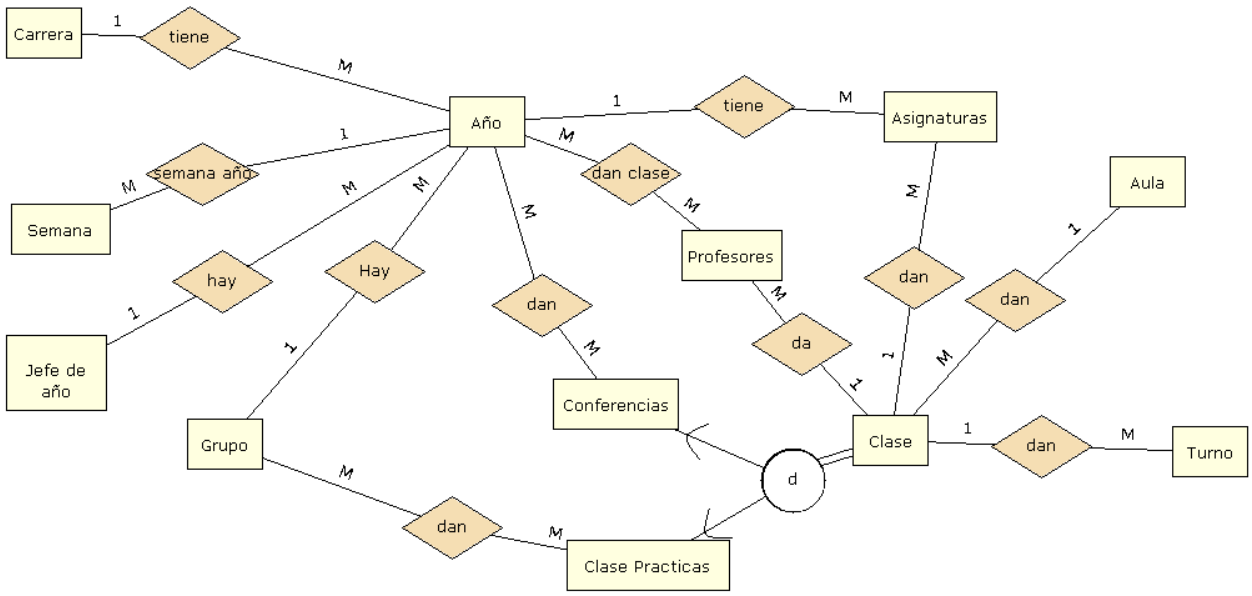


Figura 1.2. Modelo Entidad-Relación para el Sistema de Horario Docente

Los requisitos no funcionales del sistema incluyen un Servidor Web Apache 2.x, PHP versión 5.3 o superior y el sistema gestor de Base de Datos MySQL.

1.1.3 SAAC

El Sistema de Información de Apoyo a la Autoevaluación de la Carrera Ciencia de la Computación(SAAC) facilita la gestión de los datos necesarios para la elaboración de la autoevaluación semestral y de un curso académico de la carrera Ciencia de la Computación (Medina, 2013).

La figura 1.3 muestra el modelo Entidad-Relación que hace referencia a la información propia de SAAC (Medina, 2013).

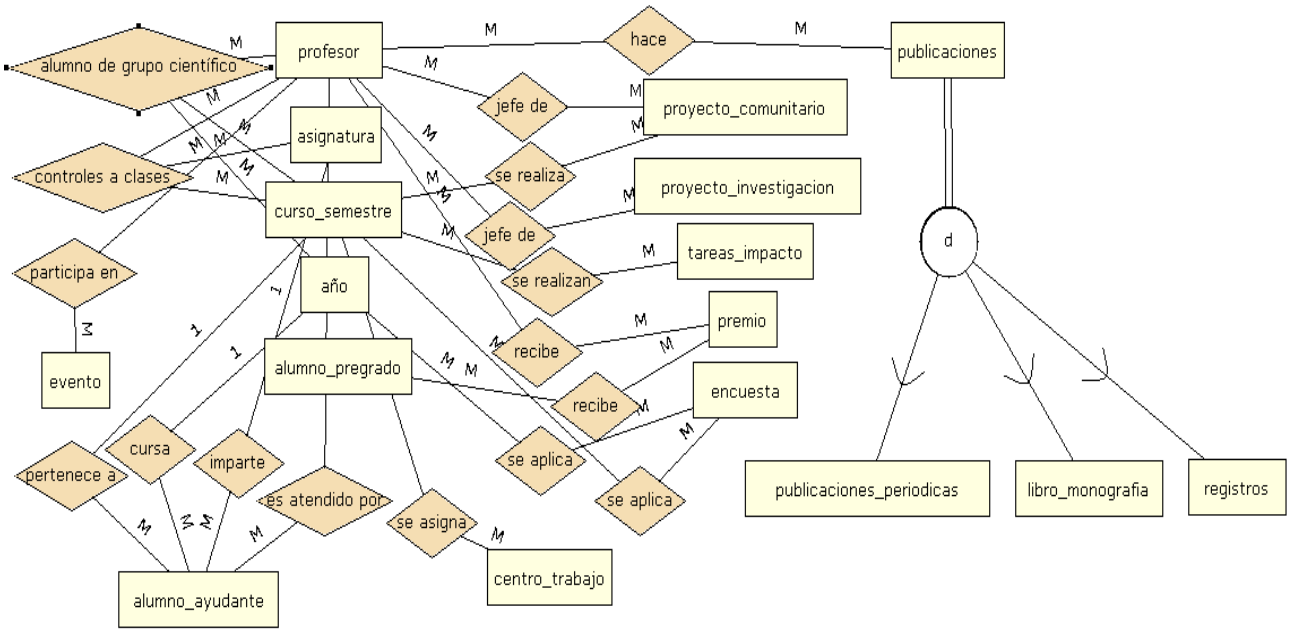


Figura 1.3. Modelo Entidad-Relación de SAAC

El sistema maneja gran volumen de información que es de vital importancia, algunos de los datos a tener en cuenta son: toda la información relacionada con profesor, los estudiantes, las asignaturas, los programas de postgrado, el claustro de la carrera y la carga docente.

SAAC requiere también de otros datos en particular: todo lo referente a proyectos comunitarios, proyectos de investigación, tareas de impacto, premios, eventos, publicaciones, resultados de las encuestas aplicadas a estudiantes, la práctica de producción y los centros de trabajos asociados a las mismas, los alumnos ayudantes, los controles a clase y las eficiencias verticales y horizontales (Medina, 2013).

SAAC es un sistema implementado en Java y se conecta a un servidor de Base de Datos PostgreSQL versión 9.0.1.

1.1.4 Sistema Control de la Carga Docente en el Dpto. Ciencia de la Computación.

El Sistema para el Control de la Carga Docente responde a la necesidad del DCC de un sistema de información que permitiera planificar y controlar las actividades en las que se encuentran involucrados los profesores del centro y que además mantenga un registro histórico de esta información, con el objetivo de emitir informes sobre la labor de los docentes a lo largo de

diferentes cursos académicos. La gestión de estas actividades comprende la carga de trabajo en pregrado y postgrado, así como la asesoría o tutoría a estudiantes (Castro, 2013).

La figura 1.4 muestra el modelo Entidad-Relación que permite exponer la información que maneja el sistema (Castro, 2013).

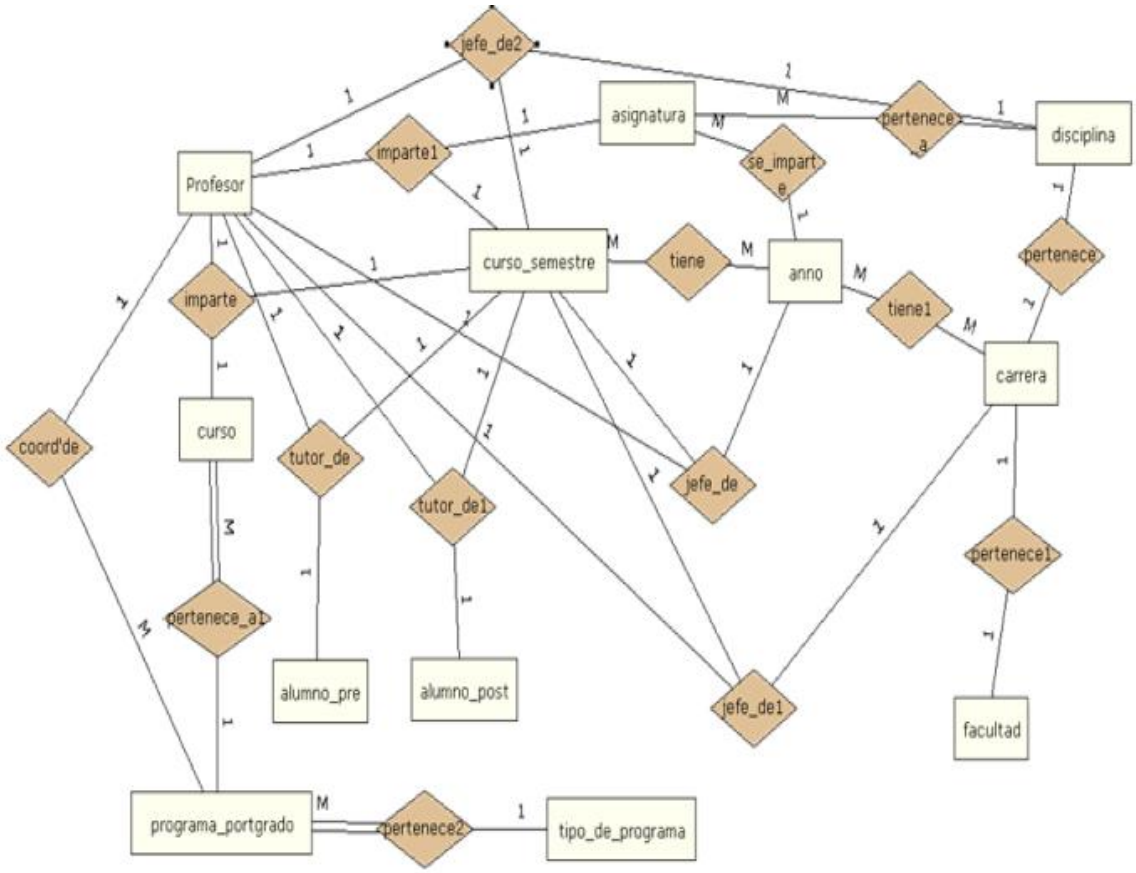


Figura 1.4. Modelo Entidad-Relación para el Sistema de Carga Docente

En la implementación del sistema se utilizó el lenguaje de programación Java y se conecta a un servidor de base de datos PostgreSQL versión 9.2.

1.1.5 Repositorio Digital Institucional del Centro de Estudios de Informática

El Centro de Estudios de Informática mantiene desde hace varios años un repositorio institucional con el objetivo de registrar y preservar datos de índole administrativo, académico y los relacionados con la producción científica de investigadores y profesores del centro.

En la construcción del repositorio se empleó la herramienta de software libre DSpace.

DSpace (http://www.dspace.org) ha sido desarrollado conjuntamente por los laboratorios Hewlett-Packard y por la biblioteca del MIT (Massachusetts Institute of Technology) específicamente para constituir servicios de repositorio institucional. Es un sistema diseñado para retener, almacenar, indexar, preservar y redistribuir en formatos digitales, la producción intelectual de una institución (Bueno de la Fuente et al., 2007).

El repositorio permite ingresar vía Web una gran variedad de recursos en formato electrónico como libros, tesis, artículos, monografías, fotografías, video, datos de investigación y otras formas de contenido.

Está diseñado para reflejar la estructura interna de la institución, de esta forma, el repositorio se estructura en *comunidades*, estas generalmente representan unidades organizativas, como departamentos académicos o administrativos, centros de investigación, laboratorios e incluso investigadores individuales. Las comunidades contienen *colecciones* que definen grupos de contenidos o de información relacionada. Cada colección está conformada por *artículos* (items), que dentro de la organización de DSpace representa los elementos registrados de un archivo o documento digital.

Cabe mencionar que las comunidades pueden organizarse en jerarquías, por lo que aquellas comunidades contenidas en otras comunidades se delimitan como *sub-comunidades.*

Teniendo en cuenta esta estructura, el repositorio de CEI está compuesto por varias comunidades. La comunidad más general (*CEI*) se divide en tres sub-comunidades (*Academia*, *Administración*, *Ciencia y Técnica*). Estas sub-comunidades tienen asociadas otras sub-comunidades y colecciones. Las colecciones agrupan los registros o documentos relacionados con la temática. En la figura 1.5 se muestra fragmentos de esta jerarquía de organización presente en el sistema.

La comunidad *Academia*, por ejemplo, tiene asociada un conjunto de sub-comunidades que representan las carreras Licenciatura en Ciencia de la Computación e Ingeniería Informática de la facultad de Matemática, Física y Computación, así como las maestrías y programa doctoral que se desarrollan en el propio centro. Dentro de estas sub-comunidades se encuentran las colecciones que representan los planes de estudios de las carreras y programas de maestría y doctorado, que almacenan los registros o documentación.

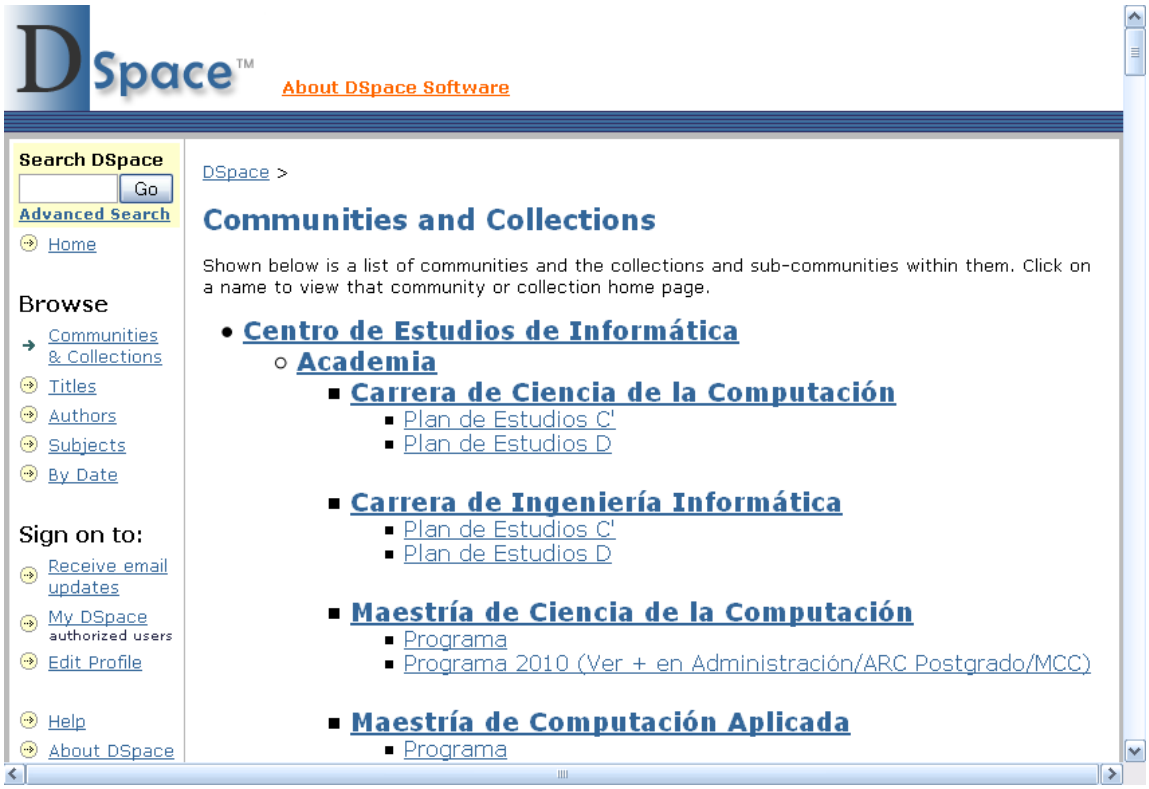


Figura 1.5. Organización del repositorio del CEI

La comunidad *Ciencia y Técnica* tiene asociado un grupo de colecciones que representan los artículos publicados por los profesores e investigadores del centro organizados por año de publicación, así como los balances de ciencia y técnica que se efectúan cada año en la propia institución. Incluye, además, una sub-comunidad denominada *Medio Ambiente* que recoge una serie de colecciones con información vinculada principalmente al trabajo del CEI en este tema.

La comunidad *Administración* contiene un volumen amplio de registros organizados en varias sub-comunidades como *ARC Aseguramiento Material*, *ARC Ciencia y Técnica, ARC Defensa y Protección, ARC Extensión Universitaria, ARC Formación del Profesional, ARC Postgrado, ARC Recursos Humanos y Cuadros, ARC Relaciones Internacionales.* Estas, a su vez, se encuentran estructuradas por conjuntos de sub-comunidades y colecciones que abarcan información de diversa índole vinculada estrechamente con el trabajo del centro.

DSpace organiza los usuarios en cuentas personales y grupos, obligando a todos los usuarios personales a pertenecer a un grupo definido para poder asignarle permisos de escritura, lectura, modificación y eliminación de registros. Estas autorizaciones pueden realizarse a nivel de comunidad o colección (Bueno de la Fuente et al., 2007). De esta forma, cada usuario registrado

estará asociado a un grupo de usuarios, donde cada grupo tendrá ciertos privilegios sobre las distintas comunidades y colecciones. Los usuarios no registrados solo podrán acceder al sistema con fines de consulta, no pudiendo registrar documentos en el repositorio.

El proceso de registro de un documento en DSpace consta de varios pasos que se inician seleccionando la colección donde se desea "colocar" el documento. Seguidamente el usuario debe ingresar una serie de datos que describen el documento a registrar. Parte de esta información se muestra en la figura 1.6.

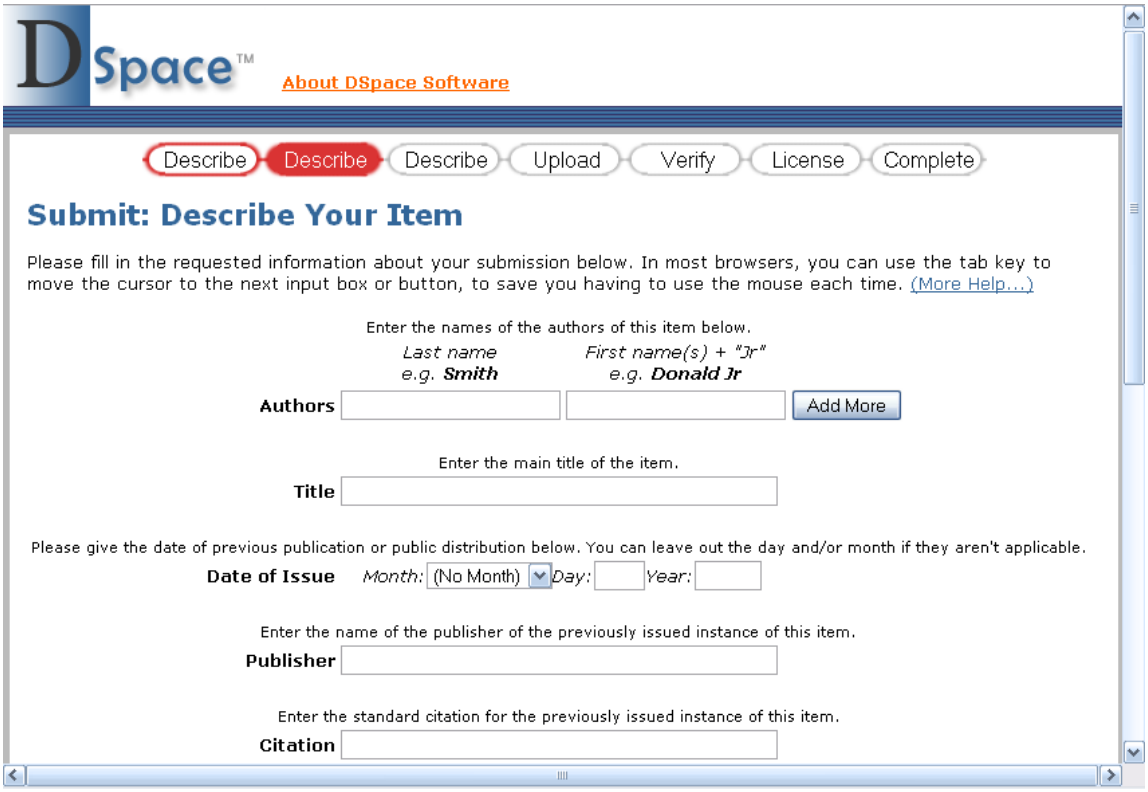


Figura 1.6. Datos del documento

El esquema de metadatos proporcionado por defecto para la descripción es Dublin Core, sin embargo múltiples esquemas de metadatos pueden ser configurados y nuevos campos de metadatos pueden ser agregados al esquema existente para describir los artículos. Actualmente el repositorio DSpace del CEI solo emplea el esquema facilitado por defecto Dublin Core para la descripción de toda la información que almacena.

DSpace establece OAI-PMH(Open Archives Initiative Protocol of Metadatas Harvesting) como protocolo de interoperabilidad (Bueno de la Fuente et al., 2007). DSpace es totalmente

compatible con el protocolo OAI-PMH. Este protocolo es una herramienta de interoperabilidad independiente de la aplicación que permite realizar el intercambio de información para que desde puntos (proveedores de servicio), se puedan hacer búsquedas que abarquen la información recopilada en distintos repositorios asociados (proveedores de datos) (Reyes Del Toro, 2007). En el epígrafe 1.4 se profundizará en este protocolo de intercambio de contenidos.

DSpace está programado en lenguaje Java y puede ser instalado tanto en sistemas Linux, como en Windows. Precisa de un servidor web Apache, el servidor de páginas dinámicas Jakarta Tomcat y Java SDK, una base de datos relacional PostgreSQL 8.0 o superior para almacenar toda la información correspondiente a la organización del contenido, los metadatos de los archivos, información sobre los usuarios y autorizaciones, permitiendo también el uso de Oracle 9.0 o posterior.

Detalles sobre otros elementos y características de DSpace pueden ser consultados en (Tansley et al., 2006) documentación de este sistema.

1.1.6 Diagnóstico del estado de informatización del CEI y la facultad MFC

Un diagnóstico es el proceso mediante el cual se lleva a cabo un análisis para recopilar información que ayude a determinar la situación actual de la organización y detectar sus áreas de mejoramiento (Febles, 2012).

Para realizar el diagnóstico de la presente tesis se efectuó un análisis de los sistemas antes mencionados, dirigido fundamentalmente a evaluar aspectos como funcionamiento, organización e información que manejan.

En el análisis realizado se pudo comprobar la existencia de duplicación de información, ante la presencia de datos dispersos por varios de los sistemas analizados, debido principalmente a que todos los departamentos utilizan sus propios sistemas de información, y estos sistemas no se encuentran conectados. La figura 1.7 muestra el escenario antes descrito y los sistemas involucrados.

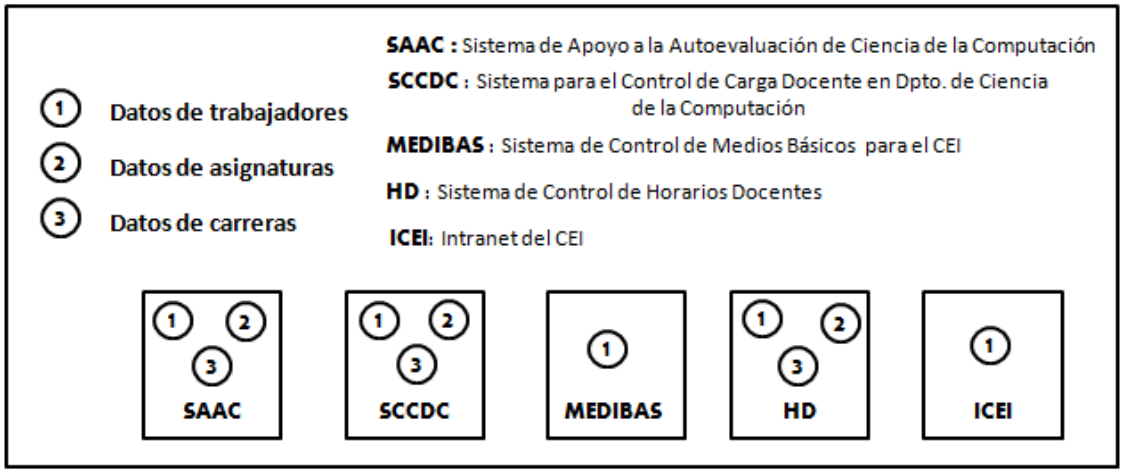


Figura 1.7. Duplicación de datos en los sistemas de información

Lo antes expuesto puede dar lugar a diferencias entre los departamentos, ya que la información que es manejada por estos sistemas no solo se almacena, sino que puede cambiar frecuentemente en los sistemas. Como consecuencia pueden presentarse problemas de inconsistencia en los datos, lo que podría conducir a dificultades en el funcionamiento de ambos centros.

Otra problemática evidenciada es la ausencia de interoperabilidad entre los sistemas estudiados. La interoperabilidad, según la definición propuesta por la IEEE es: "*la habilidad de dos o más sistemas o componentes para intercambiar información y utilizar la información intercambiada*" (Manero et al., 1990). Las causas fundamentales vienen dadas por la implementación de estos sistemas en plataformas de desarrollo diferentes, así como la ausencia de una infraestructura que facilite la comunicación entre estos sistemas.

En el caso particular del repositorio DSpace y el sitio Web de CEI la falta de interoperabilidad constituye un obstáculo ante la necesidad de acceder a la información mantenida en el repositorio con el objetivo de exponerla en la intranet del CEI.

1.2 Estrategias de integración

La creciente necesidad de acoplar elementos de software para establecer un nivel confiable de intercambio de datos ha convertido a la integración en una meta estratégica dentro del desarrollo de aplicaciones (Febles, 2012).

Según Gartner Group (Group, 2006): "La integración de sistemas significa el compartir datos y procesos de negocio en forma irrestricta entre distintas aplicaciones interconectadas…" a lo que

se puede agregar: "…De forma tal que cada proceso de negocio o porción de datos sea implementado de la mejor manera por la aplicación más adecuada, y compartido al resto"

La integración se convirtió en un punto de atención a finales de la década del 90. Numerosos sistemas fueron desarrollados con poca previsión de cómo se podrían intercambiar datos fuera del contexto del propio sistema. Como resultado se crearon soluciones de integración que provocaron ineficiencia en cuanto a la interoperabilidad, la reusabilidad y el uso de estándares en el proceso de intercambio de información entre las aplicaciones (Erl, 2007).

Las iniciativas de integración implicaban la conexión de dos o más aplicaciones o programas que pudieran ser incompatibles, estar basados en tecnologías diferentes o no estar concebidos para la conexión fuera de sus fronteras (Erl, 2007). De esta forma los enfoques para lograr la integración de aplicaciones han evolucionado con el tiempo. Estos enfoques pueden resumirse en cuatro estilos o estrategias de integración principales: Transferencia de archivos, Base de Datos compartidas, Invocación de procedimiento remota y Envió de mensajes. En los siguientes epígrafes se presentan cada una de estas estrategias.

1.2.1 Transferencia de archivos

Los esfuerzos iniciales para lograr la integración a través de múltiples aplicaciones se basaron en la transferencia de datos en archivos (Hohpe, 2002). Cada aplicación produce un archivo que contiene la información que otras aplicaciones necesitan consumir. Los archivos se convierten entonces en la interfaz pública de cada aplicación. Estas aplicaciones pueden hacer cambios internos libremente sin afectar otras aplicaciones, mientras continúen generando datos iguales en los archivos y con el mismo formato (Hohpe et al., 2004). En la figura 1.8 se representa este estilo de integración (Hohpe, 2002).

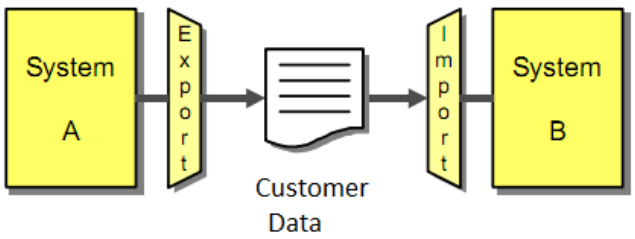


Figura 1.8. Transferencia de archivos

Una importante decisión en el uso de esta estrategia es el formato de archivo a utilizar. Como resultado, el desarrollo de formatos de archivos estándares se han incrementado con el tiempo. Un método empleado actualmente es el uso de los archivos XML.

Durante el intercambio de archivos otro punto a definir es cuándo deben ser producidos y cuándo deben ser consumidos los archivos por parte de las aplicaciones. Usualmente, se define un ciclo regular que conduce a esta decisión: todas las noches, semanalmente, mensualmente y así sucesivamente (Hohpe, 2002).

Aunque esta estrategia parece simplista, ofrece algunas ventajas. Usar archivos permite una independencia física entre los diferentes procesos. Si el sistema objetivo no está inmediatamente disponible para recibir el archivo, entonces puede ser almacenado hasta que el sistema esté listo.

Sin embargo, las transferencias de archivos también presentan una serie de desafíos. Los datos intercambiados en un sistema pueden no estar disponibles o actualizados en otro sistema hasta el día siguiente. Esto puede provocar confusión para los usuarios, y puede causar problemas de integridad de datos y de sincronización, si estos datos (pasados de tiempo) fueran actualizados en el otro sistema. Adicionalmente, estos intercambios tienden a extraer toda la información disponible de un sistema y reproducirlo para el sistema que lo necesite, si solo pocos cambios fuesen hechos a los datos, entonces esta estrategia causaría grandes cantidades de transmisiones de datos innecesarios.

1.2.2 Base de Datos compartidas

En un intento por eliminar el problema de la sincronización de los datos y la transferencia de grandes volúmenes de datos, se propuso el empleo de una base de datos compartida por todos los sistemas. Esta estrategia permite integrar aplicaciones almacenando los datos en una sola base de datos compartida y definiendo el esquema de la base de datos para manipular las necesidades de las aplicaciones (Hohpe et al., 2004). En la figura 1.9 se representa este estilo de integración (Hohpe, 2002).

Recuperar los datos de un mismo lugar traería como ventaja principal la eliminación de la necesidad de duplicación de datos. Sin embargo, el inconveniente crucial de esta solución consiste en definir un modelo de datos que se ajuste a todas las aplicaciones. También, el acceso a una sola base de datos compartida puede causar un “cuello de botella”.

Por otra parte, para esta solución, la sincronización puede ser tratada por los mecanismos de protección prevista por muchos gestores de bases de datos.

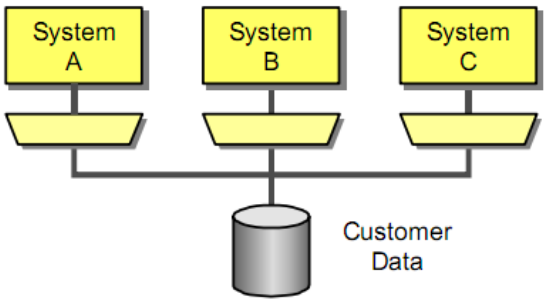


Figura 1.9. Base de Datos compartida

1.2.3 Invocación de procedimiento remoto.

El mecanismo de invocación de procedimientos remotos (conocido por sus siglas en inglés RPC) permite a una aplicación invocar de forma transparente a una función implementada en otra aplicación (Hohpe, 2002). En la figura 1.10 se representa este estilo de integración.

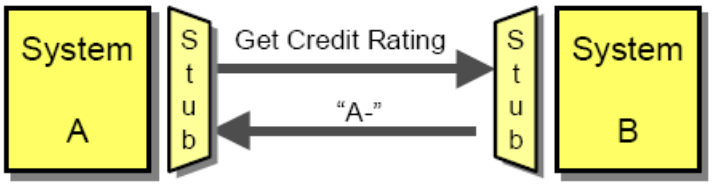


Figura 1.10. Invocación de procedimiento remoto

La invocación de procedimiento remoto aplica los principios de la encapsulación para integrar aplicaciones. Si una aplicación necesita información disponible o que posee otra aplicación, solicita directamente estos datos. Lo anterior permite que cada aplicación mantenga la integridad de los datos que posea. Además cada aplicación puede cambiar el formato de sus datos internamente sin que esto afecte a las restantes aplicaciones.

Varias tecnologías como CORBA, COM, .NET Remoting, y Java RMI implementan la invocación de procedimiento remoto (Hohpe et al., 2004).

Aunque esta estrategia concibe la integración de una forma más simple, todavía comparte los inconvenientes de una comunicación inestable. Los sistemas basados en RPC implican

conexiones punto a punto entre el remitente y el receptor, que pueden ser de difícil mantenimiento si aumenta el número de conexiones (Hohpe, 2002).

1.2.4 Envío de mensaje

El envío de mensajes intenta superar los inconvenientes de las soluciones anteriores proporcionando una transferencia de datos fiable y comunicación asíncrona (Hohpe, 2002). En la figura 1.11 se representa este estilo de integración.

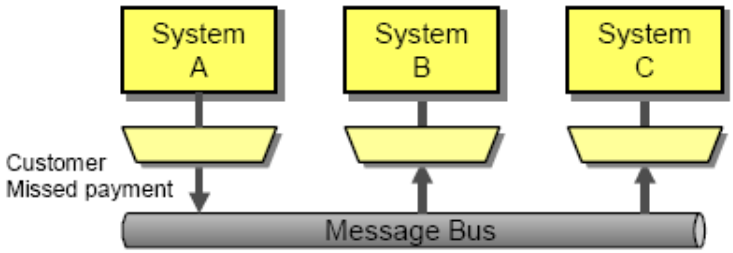


Figura 1.11. Envío de mensajes

Las aplicaciones se comunican mediante el intercambio de paquetes de datos denominados *mensajes*, empleando un sistema de envío de mensajes común. El canal de mensajes, también conocido como cola, se comporta como una colección o arreglo de mensajes, compartido por múltiples computadoras y que puede ser empleado concurrentemente por varias aplicaciones (Hohpe et al., 2004). De esta forma, la aplicación remitente transfiere el mensaje a través de canal siendo recibido por la aplicación receptora, requiriendo, por ambas partes, estar de acuerdo tanto en el canal a emplear, como en el formato del mensaje.

Algunos de los beneficios específicos de la *mensajería* incluyen (Hohpe et al., 2004):

- **Comunicación Remota**. La mensajería permite que aplicaciones independientes se comuniquen y transfieran datos.

- **Integración Plataforma/Lenguaje**. El sistema de mensajería puede ser un traductor universal entre aplicaciones que emplean diferentes lenguajes, tecnologías y plataformas, permitiendo la comunicación a través de un paradigma de mensajería común.

- **Comunicación asíncrona.** El remitente no tiene que esperar que el receptor reciba y procese el mensaje; no tiene que esperar incluso que el sistema de mensajería entregue el mensaje. El remitente sólo necesita esperar que el mensaje a enviar sea guardado con

éxito en el canal por el sistema de mensajería. Una vez el mensaje se guarda, el remitente es libre de realizar otro trabajo mientras el mensaje se transmite.

- **Regulador**. El sistema de mensajería posibilita la creación de una cola de solicitudes hasta que el receptor esté listo para procesarlos, el receptor puede entonces controlar el ritmo al que consume las solicitudes para no sobrecargarse por demasiadas solicitudes simultáneas.

- **Mediador.** El sistema de mensajería actúa como un mediador entre todas las aplicaciones que pueden enviar y pueden recibir mensajes. Una aplicación puede emplearlo como un directorio de otras aplicaciones o servicios disponibles. Si una aplicación se desconecta de las otras, necesita sólo reconectar con el sistema de mensajería, y no con todas las otras aplicaciones del sistema.

Aunque el envío de mensaje resuelve muchos de los desafíos que plantea la integración de sistemas dispares de una manera elegante, introduce también nuevos desafíos. Algunos de estos desafíos son inherentes en el modelo asíncrono, mientras otros desafíos varían con la implementación específica de un sistema de mensajería.

1.3 Patrones de integración

Los sistemas de información tradicionales tienden a proliferar como sistemas redundantes e inaccesibles entre sí (silos) que generan una gran cantidad de conexiones punto a punto (espaguetis) y provocan esquemas inconexos con marcadas ineficiencias en su funcionamiento (Davis, 2009).

La adopción de una arquitectura adecuada que localice, conecte los servicios de la TI existentes y que las nuevas aplicaciones se aprovechen de estos sin necesidad de ser construidas de cero, constituye una alternativa para eliminar las conexiones punto a punto (Febles, 2012)**.**

En este sentido, las filosofías de diseño han evolucionado y en los últimos años se han presentado renovados conceptos.

La Arquitectura Orientada a Servicios (SOA por sus siglas en inglés) es un patrón de diseño que ha revolucionado principalmente la manera de adoptar la tecnología, ha surgido como paradigma capaz de soportar los procesos de negocio, aumentando la eficacia y eficiencia de las operaciones

de las empresas en el mundo de hoy. SOA es una evolución de las arquitecturas distribuidas y los métodos, principios de integración de aplicaciones más conocidos en el mundo computacional como EAI (Enterprise Application Integration) (Febles, 2012).

1.3.1 Integración de Aplicaciones Empresariales

La Integración de Aplicaciones Empresariales (EAI), en ocasiones conocida como agente de integración, puede definirse como una combinación de procesos, software, estándares y hardware que permiten la integración de varios sistemas y posibilitan que estos se presenten como un único sistema. Otras definiciones de EAI reafirman lo antes planteado y brindan nuevas aristas:

En (Azán et al., 2009) se define EAI como el proceso de integrar múltiples aplicaciones desarrolladas independientemente, que utilizan tecnología incompatible y que son gestionadas de forma independiente, permitiendo que se comuniquen e intercambien transacciones de negocio, mensajes, y datos entre sí.

En (Schmutz et al., 2010) se plantea que el término EAI es generalmente utilizado para describir todos los métodos que intentan simplificar el proceso de hacer una conexión entre sistemas diferentes, con el objetivo de evitar un tipo de "arquitectura espaguetis", que resulta del uso incontrolado de conexiones punto a punto.

Por su parte, (Josuttis, 2007) define EAI como un enfoque para la integración de sistemas distribuidos a partir del empleo de una infraestructura común (*middleware* y/o protocolos). La infraestructura podría emplear una topología *bus* o *hub/spoke*.

Las primeras implementaciones EAI emplearon la topología conocida como *hub-and-spoke*, que consiste en una estructura centralizada que acepta las peticiones desde múltiples fuentes o aplicaciones, las cuales se conectan empleando varios adaptadores. Posteriormente, las soluciones EAI iniciaron el uso de una infraestructura de mensajes o *bus*.

La figura 1.12 describe este último estilo de EAI que posee como ventaja un enfoque descentralizado. Aplicaciones o servicios individuales pueden ser conectados al *bus* con el empleo de adaptadores (Davis, 2009).

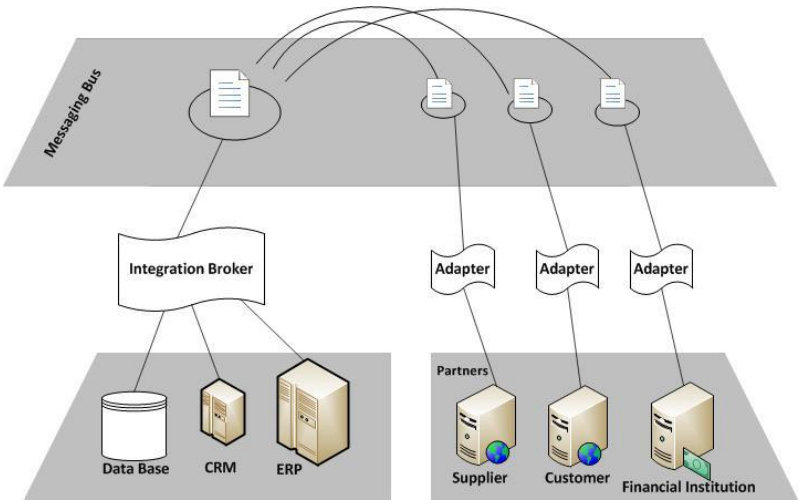


Figura 1.12. Arquitectura EAI

Desde una perspectiva del negocio, EAI puede verse como una ventaja competitiva que adquiere una organización cuando todas sus aplicaciones se integran en un sistema de información consistente. Desde una perspectiva técnica, EAI es un proceso en donde aplicaciones heterogéneas, funciones y datos son integrados, con el objetivo de permitir el uso compartido de datos y la integración de procesos de negocios desde todas las aplicaciones (Schmutz et al., 2010).

Las plataformas EAI introdujeron herramientas que permitieron una abstracción de las aplicaciones propietarias mediante el uso de adaptadores, mediadores y motores de orquestación. Estas plataformas de aplicación fueron más robustas y extensibles, pero también se convirtieron en soluciones extremadamente complejas y costosas que requerían compromisos a largo plazo con los propietarios de las plataformas de desarrollo (Erl, 2007).

La llegada de los marcos de trabajo libres para el desarrollo de servicios Web que se abstraían de la tecnología propietaria para su funcionamiento fue un nuevo horizonte en los proyectos de integración. Con ello disminuyeron las dependencias con los proveedores de las plataformas. Algunas innovaciones que alcanzaron popularidad en la era de las EAI fueron reconocidas e incorporadas a las metas globales asociadas a la orientación a servicios por la manera en que coincidían en principios como la abstracción, el bajo acoplamiento y la facilidad de composición (Febles, 2012).

EAI puede verse, entonces, como una fase preliminar de SOA, o como un concepto que compite con SOA (Schmutz et al., 2010).

1.3.2 Arquitectura Orientada a Servicios

La arquitectura orientada a servicios (Service Oriented Architecture SOA) representa un principio de diseño basado en servicios relacionados con el contexto del negocio y las TI que estimula la integración de aplicaciones independientes y busca reducir los esfuerzos en el desarrollo y mantenimiento de estas aplicaciones.

En (Josuttis, 2007) se plantea que SOA no es una arquitectura concreta, sino que, es algo que conduce a una arquitectura concreta. Puede entonces ser definido como un estilo, paradigma, concepto, perspectiva, filosofía, o representación. Es decir, SOA no es una herramienta concreta que puede ser comprada. Es un enfoque, una manera de pensar, un sistema de valores que lleva a ciertas decisiones concretas al diseñar una arquitectura de software. Mientras se aplica este paradigma a una situación concreta, se deben tomar decisiones específicas que sean apropiadas a las circunstancias.

Según IBM, es: "Una arquitectura de aplicación en la cual todas las funciones se definen como servicios independientes con interfaces invocables bien definidas, que pueden ser llamadas en secuencias definidas para formar procesos de negocios".

Un servicio es una funcionalidad concreta que puede ser descubierta en la red y que describe tanto lo que puede hacer como el modo de interactuar con él (Microsoft, 2006). En este entorno tienen su propio ciclo de vida, operan como una caja negra que puede ser accedida por diferentes fachadas como aplicaciones web, de escritorio o para móviles (Erl, 2007).

Los servicios, como parte del esquema básico de la orientación a servicios, se instrumentan en función de tres roles: el consumidor solicitante del servicio y que coincide con el cliente, el proveedor y el intermediario. Un cliente puede beneficiarse de diferentes servicios y ofertarlos como uno nuevo, convirtiéndose en proveedor. El papel del intermediario es justamente ayudar a los clientes a encontrar el servicio más adecuado a sus necesidades (Febles, 2012).

Según (Erl, 2005) el ciclo de vida de un proyecto SOA comprende una serie de pasos o fases que necesitan ser completados para construir los servicios que comprenden una solución orientada a servicios.

De esta forma, un proyecto SOA abarca las fases de: análisis y diseño orientados a servicio, implementación, prueba, despliegue y administración de los servicios. En (Erl, 2005) se define el

análisis orientado a servicios como el proceso de determinar cómo representar los requerimientos del negocio a través de la orientación a servicios. Mientras que el diseño se describe de la siguiente manera:

"El diseño orientado a servicios es el proceso mediante el cual diseños de servicios físicos concretos son derivados de los servicios lógicos candidatos y luego ensamblados en composiciones abstractas que implementan un proceso de negocio."

Las restantes fases implican la implementación de los servicios físicos a partir de su diseño, así como la comprobación rigurosa de los mismos antes de ser desplegados en el ambiente de producción.

El propio autor (Erl, 2005), plantea un conjunto de principios y buenas prácticas vinculadas con la orientación a servicios, presentes en el desarrollo y aplicación de SOA. Entre estos principios se destacan los siguientes:

- **Los servicios deben ser reutilizables**: la lógica encapsulada por el servicio debe ser suficientemente genérica como para ser empleada en numerosos escenarios y por varios consumidores.
- **Los servicios deben ser autónomos:** la lógica encapsulada por un servicio reside en un límite explícito. Los servicios tienen el control sobre este límite y no dependen de otros servicios para su ejecución. De esta manera ganan en independencia y se potencia su reutilización.
- **Los servicios deben proporcionar un contrato estandarizado**: deben expresar su propósito y capacidades a través de un contrato de servicios o interfaces técnicas mediante las cuales puedan ser accedidas y estas últimas deben responder a un estándar de definición de contrato.
- **Los servicios deben tener bajo acoplamiento:** el contrato de servicio debe ser independiente de la implementación del servicio. Los servicios deben ser independientes unos de otros.
- **Los servicios deben permitir la composición:** el diseño del servicio debe posibilitar su utilización en la construcción de otros servicios de mayor abstracción.

- **Los servicios no deben tener estado:** un servicio no debe guardar información sobre su estado y siempre debe estar listo para ser consumido en cualquier composición.
- **Los servicios deben poder ser descubiertos:** se debe poder descubrir la descripción de los servicios a fin de ser utilizado, lo que evita la creación de servicios con igual funcionalidad.

Aunque SOA es independiente de la tecnología, la plataforma tecnológica de servicios Web es una de las máximas responsables de la popularidad de SOA. Los servicios Web tienen una influencia significativa en la orientación a servicios la cual a reposicionado esta tecnología y la ha formalizado como suya propia (Erl, 2005). Muchas definiciones de SOA identifican la utilización de servicios Web en su implementación, no obstante, SOA puede ser implementado utilizando cualquier tecnología basada en servicios. En el epígrafe 1.3 se describen las principales características de los servicios Web.

Según (Febles, 2012) las razones principales para que las organizaciones decidan adoptar el paradigma orientado a servicios son: facilidad de integración de sistemas, aumenta la reusabilidad de las funcionalidades de las aplicaciones existentes, transformar y modernizar aplicaciones estratégicas, lograr respuestas ágiles a las necesidades y ganar en organización de los procesos de negocio.

De manera general, SOA supone una estrategia de organización de los elementos TI, de forma que una colección de sistemas distribuidos y aplicaciones complejas se pueda transformar en una red de recursos integrados, simplificada y sumamente flexible. Un proyecto SOA bien ejecutado permite alinear los recursos de TI de forma más directa con los objetivos del negocio, ganando así un mayor grado de integración, posibilitando la adopción de mejores decisiones y la optimización de los procesos internos y sus flujos de información, resultando en mejoras de la productividad individual. El resultado es un aumento muy notable de la agilidad de la organización.

1.3.2.1 ESB (Enterprise Service Bus)

Uno de los componentes de SOA es la infraestructura que permite utilizar los servicios. Esta es la responsabilidad del ESB, que incluye, dependiendo del enfoque técnico y organizacional previsto en su implementación, las siguientes funcionalidades (Radenakers et al., 2009):

- **Transparencia de ubicación**: El ESB permite lograr un desacoplamiento entre quien brinda una determinada funcionalidad del negocio (proveedor de servicios) y quienes desean consumirla (consumidor de servicio). De esta manera el cliente de un servicio no necesita saber dónde está ubicado el servicio que consumirá. Esto significa que una nueva localización del proveedor de servicios no tendrá impacto en el consumidor.

- **Conversión de protocolos**: El ESB permite que aplicaciones que utilizan diferentes protocolos o mecanismos de comunicación como JMS, HTTP, SOAP, «REST», FTP, SMTP, TCP, entre otros, dialoguen entre sí sin preocuparse de realizar conversiones.

- **Transformación de mensajes**: El ESB proporciona la funcionalidad para transformar mensajes de un formato a otro basado en normas como XSLT y XPath.

- **Ruteo de información**: Una funcionalidad importante del ESB es la determinación del destino final de los mensajes, ya sea mediante un ruteo basado en contenido o en reglas de negocio predefinidas.

- **Seguridad**: Los servicios de autenticación, autorización y encriptación son funcionalidades proporcionadas por el ESB para asegurar los mensajes que se envíen y prevenir el uso inadecuado de la información, con el objetivo de satisfacer los requerimientos de seguridad del proveedor de servicios. Este aspecto se detalla en el epígrafe 1.3.3.2.

- **Monitoreo**: El ESB debe ser capaz de proporcionar estadísticas de uso de sus funcionalidades que permitan validar su correcto funcionamiento.

1.3.2.2 Registros/Repositorios de servicios

La reusabilidad de los servicios en diferentes contextos es una meta fundamental de SOA. Su práctica elimina la existencia de servicios con funcionalidades similares o repetidas. Los Registros/Repositorios son las estructuras que almacenan servicios y al mismo tiempo actúan como catálogos que simplifican y automatizan la búsqueda de servicios (Erl et al., 2010).

Los proveedores publican servicios en los Registros/Repositorios para que los consumidores puedan descubrirlos, conocerlos por nombre, por funcionalidad o por propiedades específicas,

estimulando la reusabilidad y jugando un papel fundamental en la gobernabilidad de un entorno SOA (Febles, 2012).

Aunque los términos son empleados indistintamente, se ha comenzado, cada vez con más frecuencias, a emplear estos términos para indicar role diferentes (Josuttis, 2007):

- Los repositorios manejan servicios y sus artefactos, como contratos, interfaces y otros relevantes documentos que ayudan a diseñar, identificar y desarrollar servicios. Un repositorio debe contener toda la información relacionada con el funcionamiento y las interfaces de un servicio. Esta información debe ser independiente de detalles técnicos y aspectos de infraestructuras, con lo cual no debe ser necesario cambiar de repositorio cuando la organización decide emigrar hacia una nueva infraestructura (ESB).

- Los registros manejan servicios desde una perspectiva técnica. Estos manejan interfaces pero no todos los detalles del contrato: operan todos los detalles técnicos necesarios para usar un servicio. El consumidor emplea el registro para recuperar información relacionada con el servicio, como el estado de un servicio, el proveedor de un servicio y dependencias con otros servicios.

Una de los principales beneficios que ofrecen estas estructuras es planteada en (Erl, 2007) al referir que el establecimiento de Registros/Repositorios de servicios reusables y autónomos facilita la combinación de los mismos como componentes de la lógica de nuevas aplicaciones. Estas aplicaciones no son construidas desde cero, sino que son el producto de la composición del material existente implementado.

Las propiedades de las aplicaciones basadas en servicios dependen directamente de las características de estos servicios, por lo que una selección apropiada dentro de un Registro/Repositorio es de gran importancia (Febles, 2012).

1.3.2.3 SOA y Seguridad

La seguridad en SOA no difiere de la seguridad de cualquier otro sistema, por lo que comprende un conjunto de aspectos claves como (Josuttis, 2007):

- **Autentificación**: La autentificación implica la verificación de una identidad. Una identidad puede ser un usuario, un dispositivo físico, o una solicitud de servicio. En lo que respecta a SOA, esto significa verificar quién solicita el servicio.

- **Autorización**: La autorización tiene como objetivo determinar los privilegios de una identidad. En una arquitectura orientada a servicios, lo anterior comprende comprobar si la identidad que invoca al servicio posee los permisos para esta acción y/o el permiso de ver los resultados.

- **Confidencialidad**: En relación con los servicios, este aspecto permite asegurar que solo la identidad que invoca al servicio puede ver los datos del servicio mientras son transferidos entre el proveedor y el consumidor.

- **Integridad**: La clave en ese aspecto es garantizar que los datos no puedan ser manipulados o falsificados, haciendo que los mismos sean erróneos o que las credenciales de autentificación y autorización sean falsas los que implicaría una violación en el acceso a los datos.

Para la autentificación y autorización normalmente se necesitan los conceptos de IDs de usuario y contraseña, donde a cada ID de usuario generalmente le son asignados roles; privilegios como la capacidad de invocar a un servicio o la capacidad de obtener un resultado, están asociados a estos roles. Por otra parte, para la confidencialidad y la integralidad, conceptos como codificación y firma digital son empleados usualmente.

Junto con los aspectos tradicionales de seguridad, SOA introduce nuevos enfoques. Para tratar la integralidad y confidencialidad de la transferencia de datos por los servicios, la seguridad se maneja a nivel de mensaje o capa de transporte, permitiendo codificar el mensaje enviado. Otro enfoque que puede ser empleado por SOA es la de proveer la seguridad como un servicio. Un servicio de seguridad puede ofrecer a las aplicaciones la habilidad de autentificación, autorización, codificación/descodificación de los mensajes, entre otras especificaciones de seguridad. Como un servicio de seguridad es central, y no forma parte de las aplicaciones, su modelo seguridad puede desarrollarse en concordancia con las necesidades de negocio, sin afectar cualquiera de las aplicaciones.

Como parte de la arquitectura orientada a servicios pueden ser empleado diferentes estándares o normas de seguridad que permiten el manejo de estos aspectos. Algunas de los más importantes estándares son (Kanneganti et al., 2008):

- *WS-Security*.
- *WS-SecurityPolicy*.
- *WS-SecureConversation*.
- *WS-Federation*.
- *WS-Trust*
- *XML-Signature*
- *XML-Encryption*
- *WS-I Basic Security Profile*

1.3.3 Diferencias entre EAI y SOA

En las soluciones EAI se carece del concepto de servicio, así como de la reducción de complejidad y ausencia de redundancia ofrecida por los estándares abiertos. Los conceptos de servicio y de estandarización se introdujeron después con el surgimiento de la Arquitectura Orientada a Servicio que resaltó la importancia de centrarse en los niveles funcionales dentro de una compañía, y sus procesos de negocio (Schmutz et al., 2010).

De esta forma, se plantea que, emplear soluciones EAI en combinación con los servicios Web, no es equivalente a una solución SOA. Aunque el número de componentes de conexión propietarias entre los sistemas es reducido por el uso de estándares abiertos como WS-*, una solución SOA involucra un enfoque arquitectónico más extenso, basado en un perspectiva orientada a los procesos del negocio en los problemas de la integración.

Mientras las soluciones EAI se orientan a los datos y hacen énfasis en la integración de interface de las aplicaciones, SOA es un concepto orientado a procesos, donde la atención se encuentra en la integración de servicios conforme a estándares abiertos. Como resultado, se elimina la barrera entre los enfoques de integración de datos y de integración de aplicación.

La comparación entre EAI y SOA muestra varias diferencias entre los dos enfoques. En cada categoría de comparación, SOA se presenta más abierto y proporciona mayor posibilidad de

interoperabilidad sin dependencia de las plataformas propietarias. Sin embargo, la integración de sistemas y aplicaciones empleando una Arquitectura Orientada a Servicio no está libre de desafíos y limitaciones. La tabla 1.1 presenta la comparación de EAI y SOA basado en servicios Web (Pulier et al., 2006).

Tabla 1.1. Comparación entre EAI y SOA basado en servicios Web

EAI	SOA Based on Web Services
Basado en tecnología propietaria	Basado en estándares abiertos
Fuerte acoplamiento entre los sistemas	Flexible acoplamiento de los sistemas
Mínimo proveedor de interoperabilidad	La orientación a estándares provee interoperabilidad.
Reusabilidad limitada de las interfaces EAI	Interfaces de servicios muy reusables por cualquier aplicación autorizada en SOA

1.4 Servicios Web

La adopción de una solución de diseño basada en SOA, como se mencionó con anterioridad, no exige implantar servicios Web. No obstante, estos constituyen la forma más habitual de implementar SOA.

Los servicios Web son aplicaciones que utilizan estándares para el transporte, codificación y protocolo de intercambio de información, permitiendo la interoperabilidad entre sistemas de diferentes plataformas, por lo que son utilizados en una gran variedad de escenarios de integración (Microsoft, 2006).

Esta tecnología está basada en cinco especificaciones o estándares fundamentales. Dos de estos son estándares generales que existían con anterioridad al desarrollo de los servicios Web:

- *XML* es empleado como un formato general para describir modelos, formatos y tipos de datos. Permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

- *HTTP* (incluyendo HTTPS) es el protocolo de bajo nivel usado por Internet. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor. En el epígrafe 1.5 se detalla este protocolo.

Los otros tres estándares son específicos a los servicios Web y constituyen los primeros estándares especificados para esta tecnología:

- *SOAP* (Simple Object Access Protocol) especifica un mecanismo y un formato de mensaje basado en XML para intercambiar información entre aplicaciones. En el epígrafe 1.5 se aborda este protocolo.
- *WSDL* (Web Service Description Language) es el estándar definido por el W3C (World Wide Web Consortium) para describir un servicio Web y crear el contrato entre el cliente y el proveedor. Fue creado para definir la interfaz de los servicios (Febles, 2012). El documento WSDL se estructura en tres capas que definen las operaciones del servicio con los parámetros de entrada y salida junto a los tipos de datos del mensaje SOAP, internamente o referenciando un documento XSD externo, así como el protocolo y el formato por los cuales se proporcionarán las operaciones del servicio, finalizando con la localización física del propio servicio.
- *UDDI* (Universal Description Discovery and Integration) estándar de OASIS. Permite mantener un repositorio de especificaciones WSDL simplificando el descubrimiento del servicio web y el acceso a sus especificaciones (Febles, 2012).

Otro estándar relacionado con la tecnología de servicios Web es:

- *XSD* (XML Schema Definition Language): esquema que provee un lenguaje formal para la definición de la estructura y validación de documentos XML. Sirve para describir la estructura detallada de elementos que forman parte del contrato del servicio (Febles, 2012).

En el marco de desarrollo de esta tecnología se han desplegado dos tipos de servicios Web:

- Servicios basados en SOAP
- Servicios RESTful

Los servicios Web basados en SOAP son servicios donde su descripción, creación y consumo se basan formalmente en varios estándares y protocolos. En su conjunto, estos estándares son conocidos como WS-* y mantenidos por comités de regulación como W3C y OASIS. Los servicios web basados en SOAP poseen interfaces que describen las operaciones disponibles, las entradas y salidas de estas operaciones y las tecnologías a través de las cuales están disponibles las operaciones. Estas interfaces, como se mencionó anteriormente, son documentadas usando WSDL y documentos XSD (Dikmans et al., 2012).

Por su parte, los servicios RESTful están basados en un modelo muy diferente al de los servicios basados en SOAP. Este modelo es muy flexible y no requiere que el cliente conozca previamente que servicios están disponibles. La World Wide Web es un ejemplo de una implementación de arquitectura REST. Los servicios RESTful se basan en el protocolo HTTP, donde los métodos GET, POST, PUT y DELETE del protocolo son empleados para representar los tipos de operaciones Create, Read, Update, Delete (Dikmans et al., 2012).

Una de las características que diferencia ambos tipos de servicios es que mientras los servicios basados en SOAP siempre intercambian XML, los servicios RESTful no imponen restricción en el formato de los datos intercambiados, estos pueden estar en formato HTML, XML o JSON (Java Script Object Notation).

1.5 Protocolos de intercambio de información.

Los estándares y protocolos de intercambio de información constituyen mecanismos que poseen como finalidad la comunicación y transmisión de datos entre sistemas de información, posibilitando la integración, interoperabilidad, complementariedad y accesibilidad de los distintos sistemas.

Como parte de la evolución de la computación y el internet se han desarrollado varios protocolos, con características particulares, para el intercambio de información entre aplicaciones o sistemas. En los siguientes epígrafes se detallan las características principales de algunos protocolos que resultan de interés para el desarrollo de este proyecto.

1.5.1 HTTP

HTTP (Hypertext Transfer Protocol) es un protocolo del nivel de aplicación usado para la transferencia de información entre sistemas, de forma clara y rápida. Este protocolo ha sido

usado por el World Wide Web desde 1990. Está basado en otros conceptos y estándares como URI (Uniform Resource Identifier), URL (Uniform Resource Location) y URN (Uniform Resource Name) (Marshall, 2012).

El protocolo HTTP se basa en un paradigma de peticiones y respuestas. Generalmente es el cliente el que inicia la comunicación HTTP y consiste en la petición de un recurso del servidor. Esta petición puede hacerse de forma directa al servidor o a través de intermediarios.

Una petición de un cliente a un servidor ha de incluir el *método* que se aplica al recurso, el *identificador* del recurso y la *versión del protocolo* que usa para realizar la petición. Después de recibir e interpretar una petición, el servidor debe responder con un mensaje HTTP. La respuesta del servidor incluye una *línea de estado*, *cabeceras* y el *contenido del mensaje*.

La línea de estado consiste en la versión de protocolo que se utiliza, seguida de una indicación de estado numérica a la que puede ir asociada una frase explicativa. El código de estado es un número de 3 dígitos que indica si la petición ha sido atendida satisfactoriamente o no, y en caso de no haber sido atendida, indica la causa.

Se han desarrollado múltiples versiones del protocolo HTTP, muchas de las cuales son compatibles con las anteriores. El estándar RFC 2145 describe el uso de los números de versión de HTTP, donde el cliente le informa al servidor al principio de la petición la versión que emplea, el servidor utiliza entonces la misma versión o una anterior en su respuesta. Las versiones del protocolo incluyen: HTTP/0.9, HTTP/1.0, HTTP/1.1 y HTTP/1.2.

HTTP define varios métodos que indican la acción que desease efectúe sobre el recurso identificado. Los métodos más relevantes son los siguientes:

- El método OPTIONS representa una petición de información sobe las opciones de comunicación disponibles. Esto permite al cliente conocer las opciones y requisitos asociados con un recurso o las capacidades del servidor.

- El método GET requiere la devolución de información al cliente identificada por la URI.

- El método HEAD es similar al método GET, pero el servidor no tiene que devolver el cuerpo de la respuesta, sólo las cabeceras.

- El método POST es empleado para enviar datos al servidor con el objetivo de ser procesado.
- El método PUT permite guardar el contenido de la petición en el servidor bajo la URI de la petición. Si esta URI ya existe, entonces el servidor considera que esta petición proporciona una versión actualizada del recurso.
- El método DELETE se emplea para eliminar del servidor el recurso indicado por la URI de la petición.

1.5.2 SOAP

SOAP (Simple Object Access Protocol) es un protocolo cuyo principal propósito es definir un formato estándar de mensajes basados en XML que permita comunicar e intercambiar información entre componentes de software o aplicaciones a través de HTTP.

La estructura de este formato es simple, pero su capacidad para extenderse y personalizarse ha posicionado este protocolo como una de las características más significativas de SOA (Erl, 2005).

Las especificaciones SOAP fueron originalmente diseñadas para reemplazar el protocolo RPC, permitiendo que las llamadas entre componentes sean codificadas dentro de documentos XML, transportadas y luego convertidas al formato nativo del componente receptor (Erl, 2005).

Cada mensaje SOAP es empaquetado dentro de un contenedor conocido como *envelope*, el cual es responsable de encapsular las restantes estructuras del mensaje. Aquí se definen los distintos NAMESPACES que son usados en el resto del mensaje. Los NAMESPACES se utilizan para garantizar la unicidad de los elementos y evitar ambigüedades.

El mensaje puede contener un *header*, área dedicada a presentar, entre otras, información de seguridad. En muchas soluciones orientadas a servicios esta sección es parte vital de la arquitectura global, y aunque es opcional, raramente se omite. El contenido del mensaje se ofrece en el área *body*, el cual típicamente son datos en formato XML que se envían al receptor del mensaje.

Una importante característica de los mensajes SOAP es el nivel de independencia que poseen. La independencia de los mensajes se implementa a través de uso del bloque *header*. Este bloque

equipa un mensaje con toda la información requerida por cualquier servicio con el cual el mensaje entra en contacto. Lo anterior facilita la tarea a los servicios al no tener que almacenar y mantener la lógica específica de los mensajes, además de reforzar las características de reutilización e interoperabilidad presentes en la arquitectura orientada a servicios (Erl, 2005). La figura 1.13 presenta un ejemplo completo de mensaje SOAP.

Los mensajes SOAP posibilitan, además, la existencia de una sección opcional denominada *fault* que puede residir dentro de la estructura *body*, y que posee como fin almacenar un mensaje simple con información de error que será entregado en caso de ocurrir una excepción.

```
<Envelope xmlns ="http://schemas.xmlsoap.org/soap/envelope/">
  <Header>
    <x:CorrelationID
    xmlns:x="http://www.xmltc.com/tls/headersample/"
    mustUnderstand="1">
      0131858580-JDJ903KD
    </x:CorrelationID>
  </Header>
  <Body>
    <soa:book xmlns:soa="http://www.serviceoriented.ws/">
    <soa:ISBN>
      0131858580
    </soa:ISBN>
    <soa:title>
      Service-Oriented Architecture
      Concepts, Technology, and Design
    </soa:title>
    </soa:book>
    <Fault>
      ...
    </Fault>
  </Body>
</Envelope>
```

Figura 1.13. Mensaje SOAP

1.5.3 OAI-PMH

El protocolos OAI-PMH, surgió a partir de OAI (Open Archives Initiative) cuya política es la de compartir información de manera abierta.

La OAI se creó con la misión de desarrollar y promover estándares de interoperabilidad para facilitar la difusión eficiente de contenidos en Internet. Surgió como un esfuerzo para mejorar el acceso a archivos de publicaciones electrónicas, incrementando la disponibilidad de las publicaciones científicas (Barrueco et al., 2003). Sus orígenes se encuentran precisamente en el desarrollo de los repositorios, también denominados archivos, de *e-print* o ediciones preliminares

electrónicas. El mecanismo para alimentar estos repositorios era a través del depósito del autor. Las interfaces Web permitían interactuar con estos repositorios y se proporcionaban algunas herramientas de búsqueda. Se diseñaron diferentes interfaces para los distintos repositorios, por lo que los usuarios finales se veían obligados a aprender distintas interfaces para acceder a los diferentes repositorios y herramientas de búsqueda. Durante este proceso determinados actores clave llegaron a constatar que la interoperabilidad era una cuestión cada vez más importante que debía abordar la comunidad *e-print* (Carpenter, 2003). Los trabajos se centraron entonces en el desarrollo de marcos de compatibilidad para la federación de archivos de *e-prints*. Sin embargo, pronto se evidenció que permitir el intercambio de múltiples formatos bibliográficos entre distintas máquinas utilizando un protocolo común, tenían aplicaciones más allá de esta comunidad. Por ello se adoptó un objetivo mucho más amplio: abrir el acceso a un rango de materiales digitales.

Por tanto, la OAI no es solo un proyecto centrado en publicaciones científicas, sino que abarca la comunicación de metadatos sobre cualquier material almacenado en soporte electrónico. No hay nada en el protocolo que impida a los implementadores transmitir el contenido propiamente dicho de esos materiales, no obstante, se plantea que no es este el objeto principal de OAI-PMH (Barrueco et al., 2003).

Las ideas previas al surgimiento del protocolo OAI-PMH demandaban alternativas viables que proporcionaran al investigador y al usuario final la mejor opción para depositar y consultar trabajos científicos. Finalmente se propusieron dos alternativas (Martínez, 2009):

- Búsquedas simultáneas
- Recolección de metadatos

Las búsquedas simultáneas planteaban la posibilidad de explorar paralelamente diferentes servidores de datos utilizando un procedimiento determinado o algún tipo de protocolo para obtener resultados, sin embargo la velocidad del proceso de búsquedas de este tipo se ve reducida en relación a la velocidad misma de cada uno de los servidores.

La recolección de metadatos (metadata harvesting) consiste en crear una colección de recursos conformada a partir de una recopilación de metadatos obtenidos desde diferentes servidores, de esta manera el número de servidores a consultar se reduciría a uno, mejorando significativamente el rendimiento de la búsqueda. Al centrarse en un único servidor, es más sencillo crear una

interfaz de navegación venciendo la barrera de la interoperabilidad, y el proceso de ordenación de recursos se realiza de manera más eficiente.

1.5.3.1 Definición y Funcionamiento de OAI-PMH

El protocolo OAI-PMH proporciona un marco de interoperabilidad independiente de aplicaciones basado en el modelo de recolección de metadatos.

Existen dos tipos de roles en el marco de OAI-PMH (Lagoze et al., 2008):

- *Proveedores de datos* (archivos abiertos, repositorios) que manejan el depósito y la publicación de los recursos, y administran los metadatos de estos, proporcionando, a su vez, el acceso y recolección de los metadatos a través del protocolo.
- *Proveedores de servicios* que emplean los interfaces OAI de los Proveedores de Datos para recolectar y almacenar sus metadatos. Esto significa que no haya peticiones de búsqueda en directo a los Proveedores de Datos, sino que se utilizan los metadatos recolectados a través de OAI-PMH como base para la creación de servicios de valor agregado. Estos servicios pueden ser variados, sin embargo, uno de los servicios más comunes que puede ofrecer un proveedor de servicios es una interfaz de búsqueda.

Es necesario mencionar que un proveedor puede jugar los dos roles, es decir, puede funcionar como proveedor de datos y como proveedor de servicios a la vez, siendo capaz de ofrecer metadatos para ser recolectados (proveedor de datos) y ofrecer servicios para el usuario final utilizando metadatos (proveedor de servicios).

Básicamente OAI-PMH utiliza transacciones HTTP para emitir preguntas (peticiones) y obtener respuestas entre un cliente o servicio recolector de metadatos y un servidor. Los argumentos de la petición se suministran por medio de los métodos *GET* y *POST* del protocolo HTTP. OAI-PMH soporta seis tipos de petición que serán analizadas posteriormente. Las respuestas del servidor están codificadas en sintaxis XML. OAI-PMH maneja cualquier formato o estándar de metadatos codificados en XML, no obstante, Dublin Core no cualificado es el estándar de metadatos básico especificado para la interoperabilidad. En la figura 1.14 se muestra el funcionamiento básico del protocolo OAI-PMH (Carpenter, 2003).

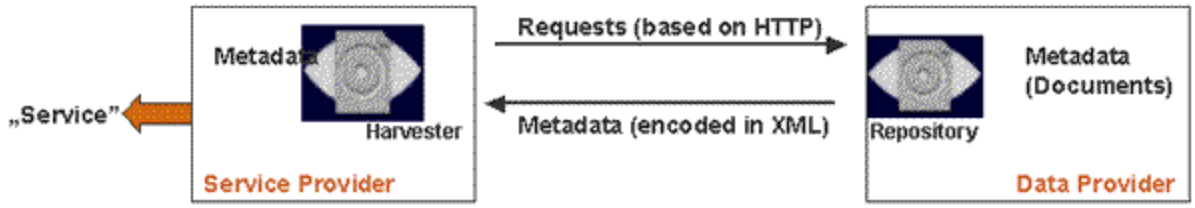


Figura 1.14. Funcionamiento básico de OAI-PMH

Resulta importante señalar que la documentación de OAI-PMH describe el uso de esquemas XML para otros formatos de metadatos como RFC 1807, para MARC 21 y para el estándar de metadatos MARC, considerando determinados factores por los cuales puede no se adecuado compartir exclusivamente Dublin Core como estándar de metadatos para un repositorio. Por ejemplo, los elementos descriptores de Dublin Core pueden ser insuficientes o no lo bastante precisos para los registros de metadatos, o puede que Dublin Core no sea el estándar de metadatos específico que se requiera. Cualquiera que sea el estándar de metadatos elegido, se debe llegar a un acuerdo sobre su uso entre los Proveedores de Datos y los Proveedores de Servicios, y se debe poner a disposición la definición de un esquema XML para su validación.

Una funcionalidad de los proveedores de datos que ha demostrado ser especialmente útiles en el funcionamiento del protocolo, dando soporte a la prestación de servicios especializados basados en la recolección selectiva de metadatos (selective harvesting) es la definición de una jerarquía lógica de *Sets*.

Los Sets definen grupos de metadatos en un repositorio, agrupándose por cualquier característica que proporcione una partición razonable para una recolección selectiva. Algunos ejemplos de *Sets* definidos por organizaciones y comunidades son las que se basan en los títulos de revistas o publicaciones seriadas, en clasificaciones o categorías temáticas, en colecciones (por ejemplo, basadas en temas o disciplinas), en tipos de recursos y en autores.

Los Sets de OAI pueden ser jerárquicos. En este caso, los Sets hijos son recolectados como parte de su Set padre. El significado de la jerarquía de *Sets* no se define en el protocolo OAI. La definición de la jerarquía de Sets puede ser interna en un repositorio, pero a menudo se fundamenta en un acuerdo entre los Proveedores de Datos o entre los Proveedores de Datos y los Proveedores de Servicios. Por último, mencionar que el soporte para construir Sets es optativo en OAI.

Otro aspecto a destacar es el tratamiento de los errores y excepciones. Los mensajes de error que maneja el protocolo se basan en HTTP. Algunos de los identificadores de error definidos son: *badArgument*, *badResumptionToken*, *badVerb*, *cannotDisseminateFormat*, *idDoesNotExist*, *noRecordsMatch*, *noMetaDataFormats* y *noSetHierarch*.

1.4.3.2 Peticiones o verbos de OAI-PMH

El protocolo OAI-PMH se compone de seis tipos de peticiones. Dentro del protocolo estas son conocidas como verbos. Estas peticiones, como se mencionó anteriormente, se realizan utilizando los métodos *GET* o *POST* del protocolo HTTP y deben proporcionar al menos un par *clave=valor*, por ejemplo, *verb=RequestType*, donde *RequestType* es algún tipo de petición. Los pares *clave=valor* adicionales dependen del tipo de petición.

Un recolector no está obligado a emplear todos los tipos de peticiones, sin embargo, un repositorio debe implementar y ser capaz de interpretar todos estos tipos.

Las peticiones que pueden realizarse al servidor son (Lagoze et al., 2008):

1. ***GetRecord*:** Es utilizado para recuperarlos metadatos de un registro concreto del repositorio. Necesita dos argumentos obligatorios, el primero es el identificador del registro pedido (identifier) y el segundo es la especificación del estándar de metadatos en que se debe devolver (metadataPrefix).
2. ***Identify*:** Es utilizado para recuperar información sobre el repositorio: nombre, versión del protocolo que utiliza, dirección del administrador, etc.
3. ***ListRecords*:** Recupera registros de un repositorio. Los parámetros posibles son los mismos que para el verbo *ListIdentifiers.*
4. ***ListIdentifiers*:** Forma abreviada de *ListRecords,* recupera solo los encabezamientos de los registros, en lugar de los registros completos. Recibe cinco argumentos, solo uno obligatorio, la especificación del estándar de metadatos en que se debe devolver (metadataPrefix). Otros argumentos son opcionales y permiten, por ejemplo, especificar el rango de fechas entre las que queremos recuperar los datos (*Set*).
5. ***ListSets*:** Recupera la estructura de *Sets* de un repositorio. Recibe un argumento resumptionToken con un valor que identifica el "control de flujo" y que fue retornado por una petición previa de *ListSet* que emitió una lista incompleta de resultados.

6. ***ListMetadataFormats*:** Devuelve la lista de estándar de metadatos disponibles del servidor. Permite un argumento opcional (identifier).

1.5.3.3 Flexibilidad de implementación

OAI-PMH permite ser implementado de distintas maneras dependiendo de los objetivos y necesidades de cada caso. Permite ser implementado de manera flexible dado que es un protocolo basado en HTTP y XML, tecnologías totalmente compatible con la Web.

A continuación se muestran algunos diagramas que muestran diferentes formas en que sistemas basados en el protocolo OAI-PMH pueden configurarse.

Entre los diferentes tipos de implementación del protocolo una de las más comunes es aquella en la que varios proveedores de servicios pueden recolectar metadatos de varios proveedores de datos como se muestra en la figura 1.15 (Carpenter, 2003).

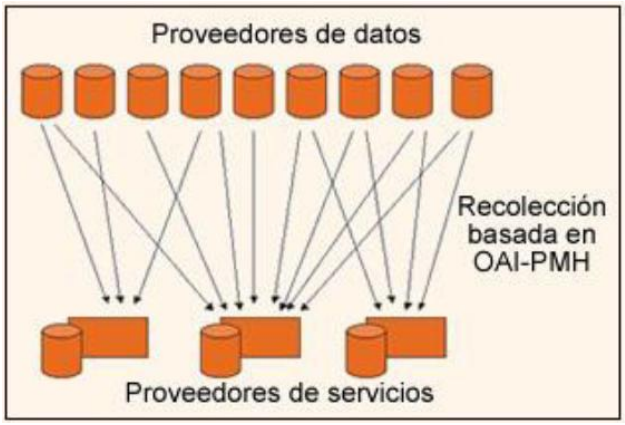


Figura 1.15. Implementación más común de OAI-PMH

Otro tipo de implementación es aquella en la que un *agregador* se sitúa de manera intermedia entre proveedores de datos y proveedores de servicios actuando de ambas maneras. Un *agregador* OAI es un servicio que reúne metadatos de varios Proveedores de Datos y después los pone a disposición de otros empleando OAI-PMH. En este caso, los Proveedores de Servicios deben conocer la identidad de los Proveedores de Datos que han sido agregados. Esto permite a los Proveedores de Servicios evitar los duplicados que pudieran surgir en la recolección del *agregador* y de los Proveedores de Datos originales. En la figura 1.16 se observa esta implementación (Carpenter, 2003).

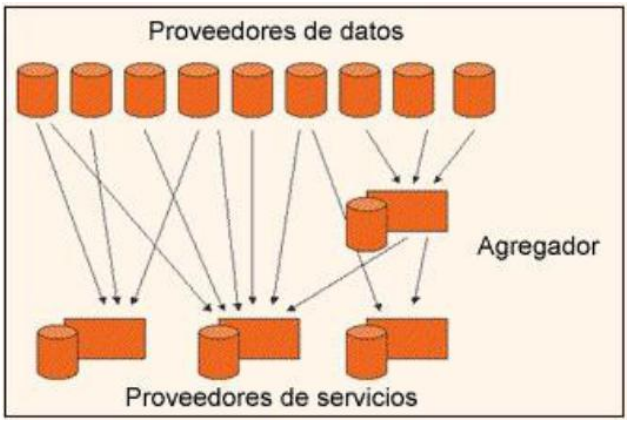


Figura 1.16. Implementación con un agregador de servicios intermedio

La tercera implementación y la más rara de encontrar en un sistema funcional real, debido a que pueden presentarse problemas de incompatibilidad, es aquella donde se combinan diferentes protocolos de transmisión de contenidos. En la figura 1.17 se observa esta implementación (Carpenter, 2003).

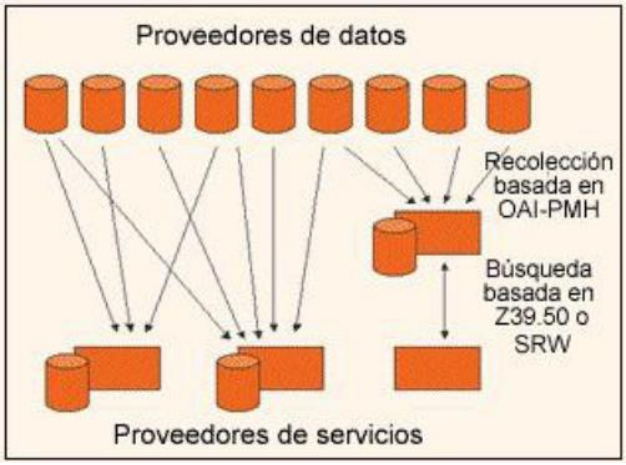


Figura 1.17. Combinación de diferentes protocolos

La flexibilidad del protocolo OAI-PMH ofrece varias posibilidades de implementación.

1.5.3.4 Versiones de OAI-PMH

Para definir una versión del protocolo OAI-PMH los expertos de la OAI debatieron y expusieron diferentes alternativas y puntos de vistas con el objetivo de encontrar la manera más eficiente de facilitar la recolección de metadatos entre diferentes proveedores de internet. Se fijaron cuatro elementos fundamentales para una primera versión del protocolo (Martínez, 2009):

- Un protocolo de transporte (HTTP o FTP)
- Un formato de metadatos MARC o Dublín Core)
- Una base para el control de calidad de los metadatos (conjunto de elementos obligatorios, convenciones sobre nombres y materias, etc).
- Propiedades intelectuales y derechos de uso (¿quién puede hacer qué con qué?)

Esta primera versión fue llamada "Convenio de Santa Fe", haciendo referencia al lugar de encuentro de los expertos donde se consolidó este acuerdo.

La "Convenio de Santa Fe" fue la primera versión implementada del protocolo OAI-PMH y tenía como objetivo primordial mejorar las búsquedas de material científico autoarchivados por los mismos autores (Barrueco et al., 2003).

El siguiente paso en el desarrollo del protocolo fue la versión 1.0 que incluyó, entre sus principales mejoras, la implementación de Dublin Core no cualificado como el estándar de metadatos mínimo para garantizar la interoperabilidad entre metadatos a transmitir utilizando este protocolo. Otra diferencia respecto a la primera versión del mismo, fue un cambio en el enfoque para facilitar no sólo la búsqueda de material autoarchivado por los autores, sino también de todos aquellos "objetos tipos documentos". Se trataba de una especificación de interoperabilidad sencilla que pretendía únicamente ofrecer una alternativa para la recolección de metadatos, por lo cual, no proporcionaba soporte para búsquedas; es decir, se enfocaba solamente al aspecto de recolección de metadatos entre los repositorios de los proveedores de datos en Internet. A partir de esta versión se usó ya el nombre oficial del protocolo: OAI-PMH.

La versión OAI-PMH 1.1 no fue más que una corrección de la versión 1.0 que tenía en cuenta los cambios en la nueva especificación del esquema XML. Cabe mencionar que tanto la versión 1.0 y la versión 1.1 fueron solamente de propósito experimental.

La versión OAI-PMH 2.0 implicó una revisión más intensa y minuciosa de las versiones anteriores, tal fue la intensidad y corrección respecto a las versiones anteriores, que la versión 2.0 no es compatible con ellas. Una vez más se amplió el enfoque del protocolo, planteando que se trata de "el intercambio recurrente de metadatos de recursos entre distintos sistemas". La versión 2.0 del protocolo sigue siendo una especificación de interoperabilidad de bajo nivel basado en la recolección de metadatos en Internet (Martínez, 2009).

A partir de la versión 2.0 el protocolo OAI-PMH deja de ser una versión experimental y se convierte en una versión estable.

1.6 Conclusiones del capítulo

Después de un estudio sobre el estado actual de la informatización en el CEI y la facultad de MFC que condujo a la identificación de un conjunto de problemas y al análisis del estado del arte referido a la aplicación de patrones y estándares en la integración de aplicaciones, se concluye que:

- Las principales problemáticas que enfrentan las aplicaciones que actualmente se despliegan en el CEI y la Facultad de MFC se centran en la duplicación de datos y de funcionalidades, y en la falta de interoperabilidad entre las aplicaciones. Esto puede conducir a inconsistencia e inflexibilidad entre la aplicaciones En el caso particular del sitio web del CEI estos problemas impiden la implementación de una funcionalidad importante, la exposición desde sus páginas del currículo de los profesores e investigadores.
- Existen varios modelos que permiten eliminar la estructuración aislada e independiente de las TI. Cada modelo ofrece criterios y características propias, y de su adopción adecuada depende la mejora en la estructuración de las aplicaciones que facilite su comunicación, y por tanto la reutilización de funcionalidades y datos, y brinde agilidad ante nuevos cambios.
- Por las características y las posibilidades que ofrece la Arquitectura Orientada a Servicios se propone su empleo con el objetivo de dar solución a los principales problemas presentes entre las aplicaciones y posibilitar la inserción de nuevas aplicaciones en el marco de esta arquitectura.
- Al presentarse un escenario particular de integración y analizando las características y funciones específicas que ofrece OAI-PMH, se propone el empleo de este protocolo para el logro de la interoperabilidad entre el sitio web del CEI y el repositorio DSpace, lo que permitirá el acceso y recuperación de la información almacenada en el repositorio para su posterior empleo desde el sitio web.

CAPÍTULO 2. Análisis y diseño de una solución SOA.

En el capítulo se presenta el análisis y diseño de la Arquitectura Orientada a Servicio, propuesta que permitirá la integración de los principales sistemas que se encuentran en explotación en la facultad MFC y el CEI. Se abordan aspectos necesarios para el desarrollo de dicha arquitectura, presentando el análisis y descripción de los servicios que integran la arquitectura. Se presentan herramientas que permiten el diseño y la implementación de los servicios.

2.1 Análisis orientado a servicios.

La Arquitectura Orientada a Servicios, como se ha mencionado, es un patrón de diseño para desarrollar una infraestructura de TI a partir de componentes acoplados de manera flexible, denominados servicios, que desempeñan funciones específicas.

La adopción de una arquitectura SOA no significa descartar la estructura y el funcionamiento de las TI que están siendo empleadas para remplazarlas con una nueva arquitectura, tampoco significa desarrollar la TI completamente nuevas. El proceso de adopción de una arquitectura SOA se dirige hacia la renovación de las TI, tomando como partida la organización ya existente. Adicionalmente, el desarrollo de una solución SOA, aún cuando se inicie con un proyecto pequeño, debe dejar abiertas las posibilidades a proyectos más grandes que pudieran ser ampliados en un futuro.

Teniendo en cuenta lo antes planteado y considerando los escenarios analizados durante el estudio de los sistemas que se encuentran actualmente en funcionamiento en la facultad de MFC y el CEI, en la fase de análisis se identificaron un conjunto de servicios candidatos, que permitieron la determinación de los servicios concretos en la fase de diseño. Estos servicios tienen como propósito lograr la comunicación y sincronización entre los sistemas, permitiendo el intercambio de información con el objetivo de eliminar los problemas de duplicación de datos y funcionalidades, esto conduce a cambios dentro de las estructuras internas de los sistemas.

Entre los cambios que se proponen está la eliminación de la información de trabajadores que se maneja en los sistemas: Intranet del CEI, el Sistema de Control de Medios Básicos para el CEI, el Sistema de Control de Horario Docentes, el Sistema de Información de Apoyo a la Autoevaluación de la Carrera Ciencia de la Computación y el Sistema para el Control de la

Carga Docente en el Departamento de Ciencia de la Computación. La presencia de estos datos en cada uno de los sistemas provoca la duplicación de información.

Para poner en práctica estas modificaciones es necesaria la creación de una base de datos común que almacene la información de los trabajadores docentes y no docentes. La figura 2.1 presenta el diseño de esta base de datos, que a los efectos de esta tesis se denominará TRABAJADORES. La llave primaria de la tabla *trabajador* de esta base de datos corresponderá con el número del carnet de identidad de cada trabajador (*ci*).

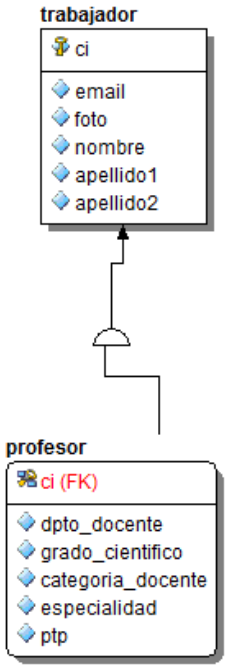


Figura 2.1. Esquema lógico de la base de datos Trabajadores.

De las bases de datos de los sistemas antes mencionados se deben eliminar las tablas que recogen los datos de los trabajadores. De esta forma, en el Sistema de Información de Apoyo a la Autoevaluación de la Carrera Ciencia de la Computación y en el Sistema para el Control de la Carga Docente en el Departamento de Ciencia de la Computación se elimina la tabla con nombre *profesor*. Mientras que en el Sistema de Control de Horario Docentes se elimina la tabla *profesores*. De igual forma en el Sistema de Control de Medios Básicos para el CEI se realiza la eliminación de la tabla *trabajador* y en la base de datos del Sitio Web del CEI se elimina la tabla *uclv_profesor*.

Varios aspectos se deben tener en consideración antes de la eliminación de estas tablas:

1. Mantener referencias desde las bases de datos de los sistemas a la base de datos común TRABAJADORES. De este modo, las tablas que mantienen relaciones de integridad referencial con la tabla que representa a trabajadores o profesores dentro de las bases de datos de los sistemas, conservarán identificadores que referenciarán a cada trabajador en la base de datos común.
2. Garantizar la sincronización de estos identificadores entre todos los sistemas con la llave primaria de la tabla *trabajador* de la base de datos TRABAJADORES, con el objetivo de prevenir el riesgo de inconsistencias entre los identificadores. Para el logro de este requisito se propone emplear como identificador común entre todos los sistemas el número de carnet de identidad (*ci*) del trabajador.
3. Actualmente, en las tablas de las bases de datos que recogen la información de los trabajadores en cada uno de los sistemas, se emplea un valor numérico para identificar unívocamente a cada trabajador. Para asegurar que el cambio de los identificadores por el nuevo valor de *ci* se produzca en todas las tablas de la base de datos que tengan relaciones de integridad referencial con la tabla que representa a los trabajadores, se debe habilitar en cada relación de estas tablas la actualización en cascada. Para esto se emplea la sentencia:

   ```
   ALTER TABLE [CHILD_TABLE] ADD CONSTRAINT
   [NAME_RELATION] FOREIGN KEY
    (ID_FK) REFERENCES [PARENT_TABLE] (ID_PK) ON UPDATE CASCADE
   ```

4. Posterior a la actualización del identificador se debe deshabilitar las relaciones de las tablas con la tabla que recoge la información de trabajadores.

Una vez realizados los cambios anteriormente analizados, se puede proceder a eliminar de las bases de datos de cada sistema mencionado las tablas que almacenan los datos de trabajadores.

A continuación se presenta, como forma de ejemplo, el diseño de la base de datos del Sistema de Control de Medios Básicos para el CEI, posterior a la realización de los cambios propuestos. En la figura 2.2 se muestra el esquema físico del sistema con los cambios realizados.

Atendiendo a los pasos descritos anteriormente se habilita la actualización en cascada, mediante las sentencias:

ALTER TABLE dbo. trabajador ADD CONSTRAINT
trabajador_cuenta FOREIGN KEY
(id_trabajador) REFERENCES dbo.cuenta (id_trabajador) ON UPDATE CASCADE

ALTER TABLE dbo. trabajador ADD CONSTRAINT
trabajador_medio_basico FOREIGN KEY
(id_trabajador) REFERENCES dbo.medio_basico (id_trabajador) ON UPDATE CASCADE

Se ha eliminado la tabla *trabajador* y por tanto las referencias que existían desde *cuenta* y *medio_básico*. En su lugar se mantiene el identificador *ci* que permitirá referenciar a la base de datos TRABAJADORES.

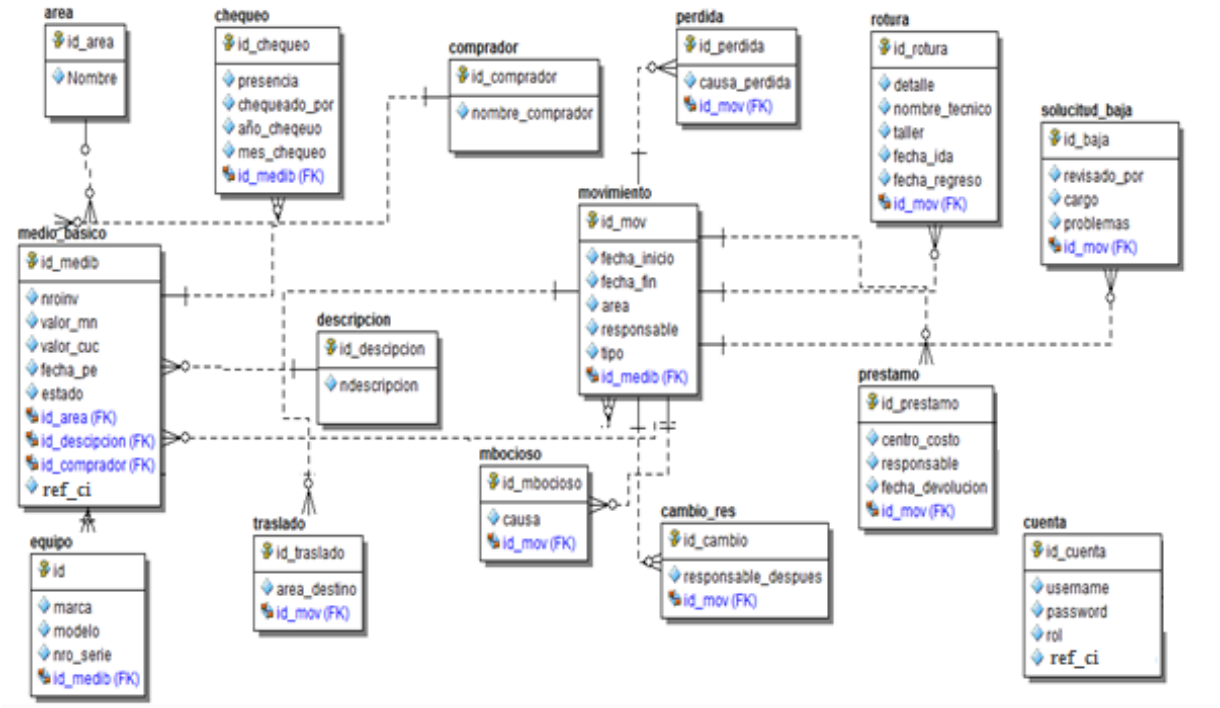


Figura 2.2. Nuevo diseño físico de la base de datos del Sistema de Control de los Medios Básicos del CEI

Las operaciones de inserción, eliminación, actualización y recuperación de los datos de trabajadores, que forman parte de los requisitos funcionales de los sistemas, se efectuarán accediendo a los servicios identificados que implementarán estas funciones. Los permisos para efectuar estas operaciones en la base de datos TRABAJADORES se corresponden con los permisos asignados a cada usuario que interactúa con los sistemas involucrados.

Por otra parte, los datos de asignaturas y carreras aparecen también en varios de los sistemas analizados. En este caso se propone realizar cambios en el Sistema de Control de Horario Docentes, eliminando de este sistema la información referente a asignaturas y carreras, datos estos que deberán ser obtenidos de la bases de datos del Sistema de Información de Apoyo a la Autoevaluación de la Carrera Ciencia de la Computación y el Sistema para el Control de la Carga Docente en el Departamento de Ciencia de la Computación.

De igual forma que en el caso analizado anteriormente, se propone el empleo de identificadores en la base de datos del Sistema de Control de Horario Docentes para referenciar la información de carreras y asignaturas en la base de datos de los sistemas SAAC y SCCDC.

En el momento de realizar estos cambios, se debe tener en cuenta que:

1. Antes de eliminar de la base de datos del Sistema de Control de Horario Docentes las tablas que recogen la información referente a carreras y asignaturas (tablas *Asignaturas* y *Carrera*), se debe habilitar la actualización en cascada para garantizar que los identificadores en las tablas *Clase* y *Año*, que antes referenciaban a las tablas *Asignaturas* y *Carrera*, se sincronicen con los identificadores en SAAC y SCCDC, con el propósito de mantener la integridad y lograr el correcto funcionamiento de la operaciones entre los sistemas.
2. Luego de deshabilitar las relaciones de integridad referencial de las tablas *Asignaturas* y *Carrera* se procede a sus eliminaciones.

La gestión de los datos desde los sistemas SAAC y SCCDC se posibilitará mediante los servicios que exponen estas funcionalidades.

En la figura 2.3 se representa el escenario descrito hasta el momento, incluyendo algunos de los servicios candidatos identificados que apoyarán la realización de las modificaciones propuestas.

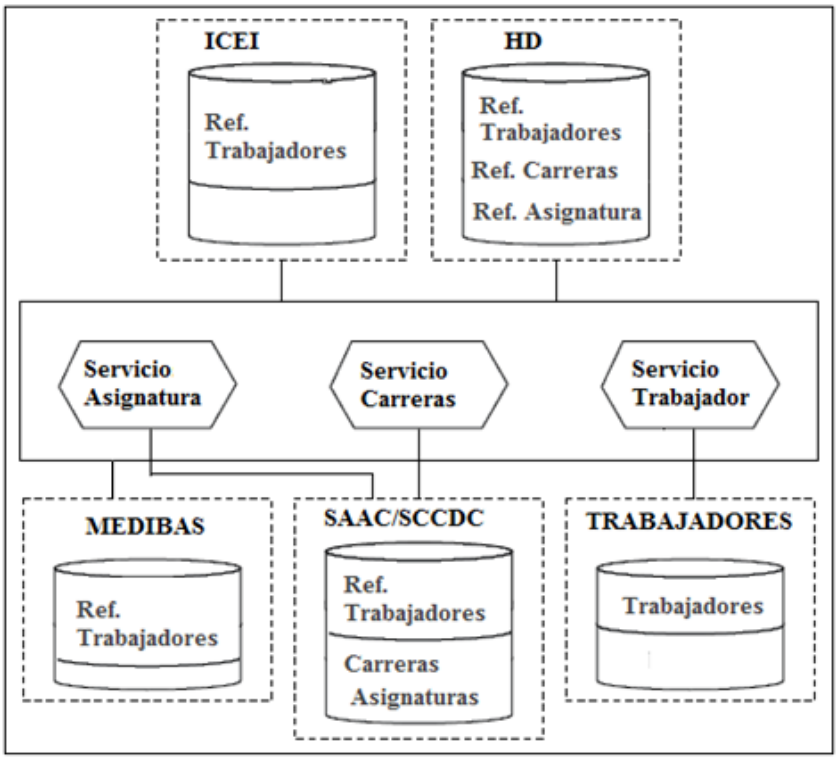


Figura 2.3. Modificaciones en la estructura interna de las bases de datos.

El análisis realizado permitió la identificación de un servicio necesario, que permita mantener la sincronización y la integridad de los identificadores entre los diferentes sistemas luego de efectuar las operaciones de inserción, eliminación y actualización de información relacionada con trabajadores, asignaturas y carreras desde los sistemas involucrados.

2.2 Diseño orientado a servicios.

Concluida la fase de análisis, se inicia la fase de diseño que incluye la descripción de los servicios y donde se propone la infraestructura que sostendrá la solución SOA. La figura 2.4 presenta la propuesta de la arquitectura orientada a servicios que permitirá la integración de los sistemas del CEI y de la facultad de MFC.

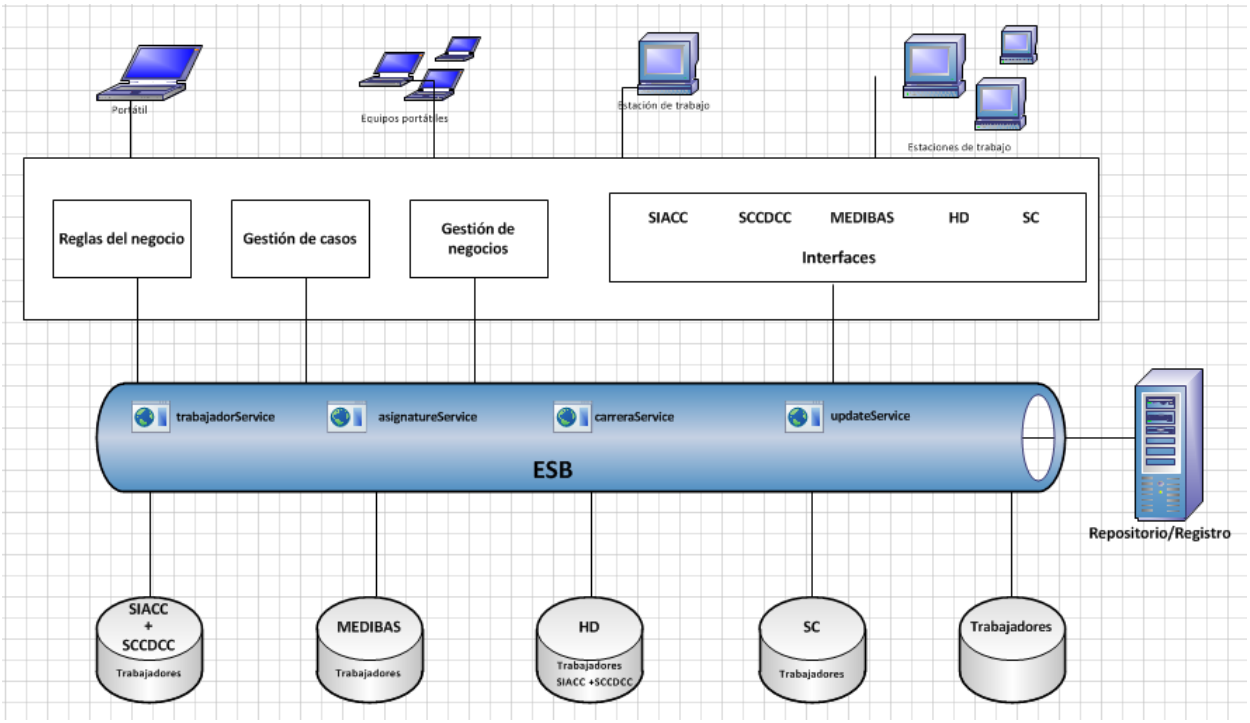


Figura 2.4. Diseño de la arquitectura SOA.

Los componentes principales que conformarán la infraestructura serán:

Registro/Repositorio: Se propone como estructura imprescindible en el almacenamiento y control de los servicios, siguiendo un estándar UDDI. Estas estructuras almacenarán el WSDL, XSD, políticas, y especificaciones técnicas de los servicios. Permitirá gestionar el ciclo de vida de los servicios así como su descubrimiento.

Bus de servicios (ESB): Permite la transparencia de ubicación, la transformación, enriquecimiento y ruteo de mensajes, el monitoreo de las comunicaciones, el manejo de eventos y diferentes mecanismos de seguridad. Proporciona la facilidad de que si un equipo falla, o si cambia la ubicación de un proveedor de servicio, no es preciso notificar el cambio a cada uno de los clientes individuales. Esto puede contribuir significativamente a la reducción de los costos de gestión de las TI y a minimizar los riesgos.

2.2.1 Descripción de los servicios.

Los servicios pueden ser clasificados en tres categorías (Josuttis, 2007):

- **Servicios básicos**: Cada servicio provee una funcionalidad básica dentro del proceso del negocio.

- **Servicios compuestos:** Estos servicios se componen por otros servicios básicos y/o por otros servicios compuestos.
- **Servicios de proceso**: Representan flujos de trabajo o procesos del negocio.

Existen dos tipos de servicios básicos:

- **Servicios básicos de datos**: Leen o escriben datos desde o hacia un sistema de almacenamiento. Estos servicios típicamente representan operaciones fundamentales del sistema.
- **Servicios básicos lógicos**: Representan las reglas de negocio fundamentales. Estos servicios normalmente procesan algunos datos de entrada y retornan los resultados correspondientes.

El proceso de análisis permitió identificar las características siguientes:

- Funcionalidades que se desean exponer.
- Datos de entrada y de salida de cada una de las funcionalidades.
- Forma de agrupar las funcionalidades.

Con esta información se puede deducir que las funcionalidades serán las operaciones de servicios, y que la agrupación de dichas funcionalidades, siguiendo algún lineamiento, serán los servicios. Los datos de entrada y de salida permitirán conformar los mensajes que los contendrán.

Analizando los servicios candidatos identificados, y teniendo en cuenta los principios y buenas prácticas vinculados a la orientación a servicios, referenciados en la literatura, se determinaron los servicios concretos a diseñar para implementar la solución SOA, que se clasifican como servicios de datos.

En la implementación de la solución SOA se propone la utilización de servicios Web, basándose en el estándar SOAP para el intercambio de información. Los contratos de los servicios serán expuestos por el estándar WSDL.

El estándar WSDL, como se mencionó anteriormente, permite tener una descripción de los servicios Web. Especifica la interfaz abstracta a través de la cual un cliente puede acceder al servicio y a los detalles de cómo puede ser empleado.

La estructura del WSDL tiene los siguientes elementos:

- < *definitions* >: Es la raíz o elemento padre del documento WSDL.
- <*types*>: Esta sección define los tipos de datos usados en los mensajes. La definición de los tipos puede hacerse directamente en esta sección o referenciando a un documento XSD externo.
- <*message*>: Este elemento define los nombres de los mensajes. Contiene el elemento hijo <*part*> que asigna un tipo a cada mensaje (petición o respuesta). Luego cada mensaje es asociado a una operación en el elemento <*portType*> para definir las entradas y salidas de las operaciones.
- <*portType*>: Con este apartado se definen las operaciones del servicio.
- <*binding*>: Se especifica un protocolo de comunicación que puede ser usado para acceder e interactuar con el WSDL.
- <*service*>: Provee una dirección física a partir de la cual se puede acceder al servicio.
- <*documentation*>: Un elemento opcional que permite incorporar descripciones o aclaraciones dentro de la definición del WSDL.

A continuación se describen los servicios identificados que se proponen sean implementados como componentes principales de SOA. Los contratos de los servicios fueron diseñados de una manera estandarizada, atendiendo a una de las mejores prácticas en el desarrollo de servicios (Contract First) que propone el desarrollo de los documentos WSDL antes de la lógica de los servicios, lo que apoyará la posterior implementación de estos, y teniendo como apoyo el WSO2 Developer Studio, basado en Eclipse IDE (Integrated Development Environment). Posteriormente se validaron los contratos como se muestra en el Anexo 1.

El servicio nombrado *serviceTrabajador* será el encargado de realizar las operaciones relacionadas con la inserción, recuperación, eliminación y actualización de los datos de trabajadores docentes y no docentes en la base de datos común TRABAJADORES. La figura 2.5 muestra el diseño de este servicio con sus operaciones.

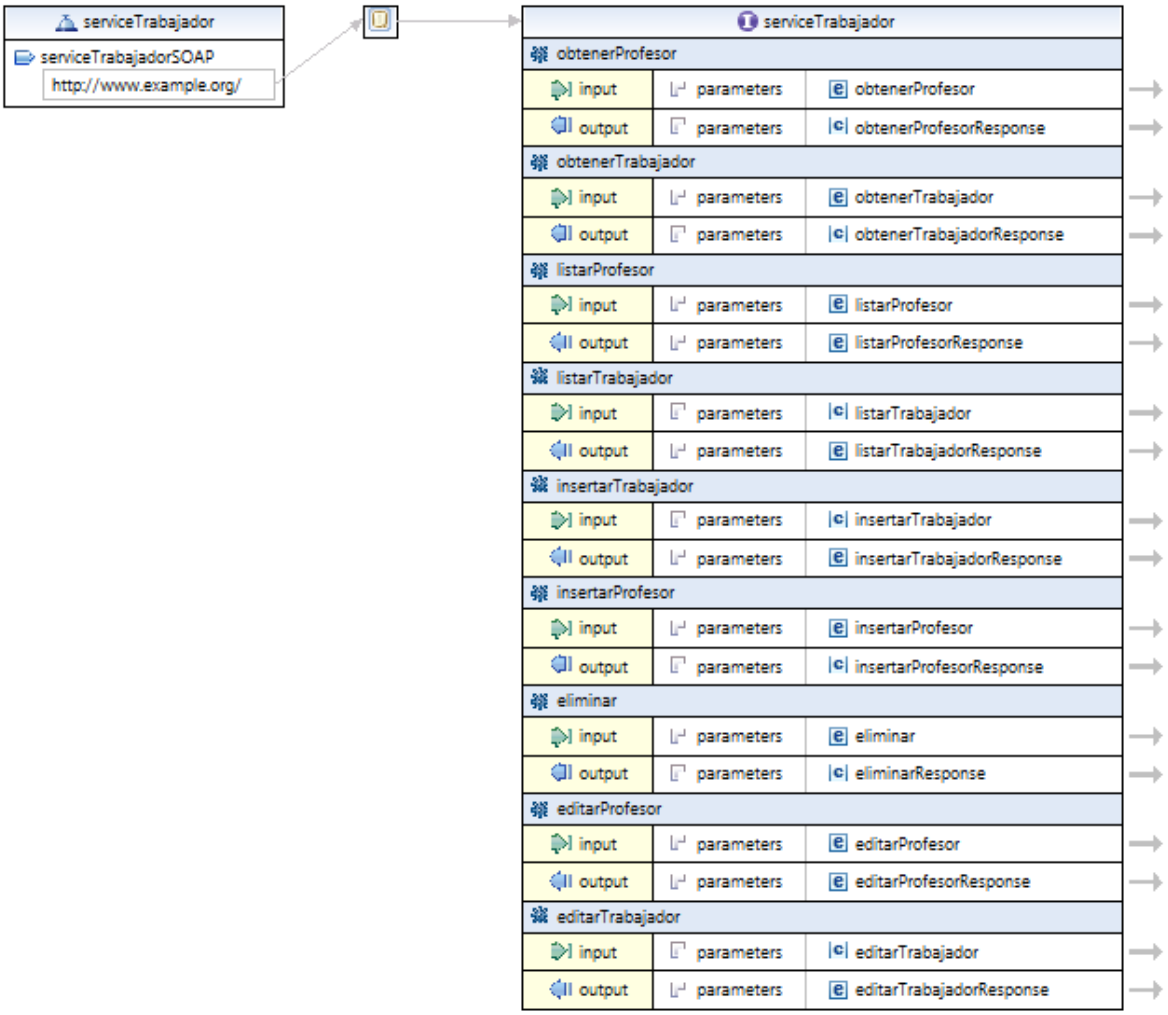


Figura 2.5. Representación de *serviceTrabajador*

La operación *obtenerProfesor* permite la recuperación de los datos de los trabajadores docentes una vez que se haya solicitado. Esta operación recibe como parámetro el identificador de trabajador (*ci*). La figura 2.6 presenta los parámetros de entrada y salida de la operación.

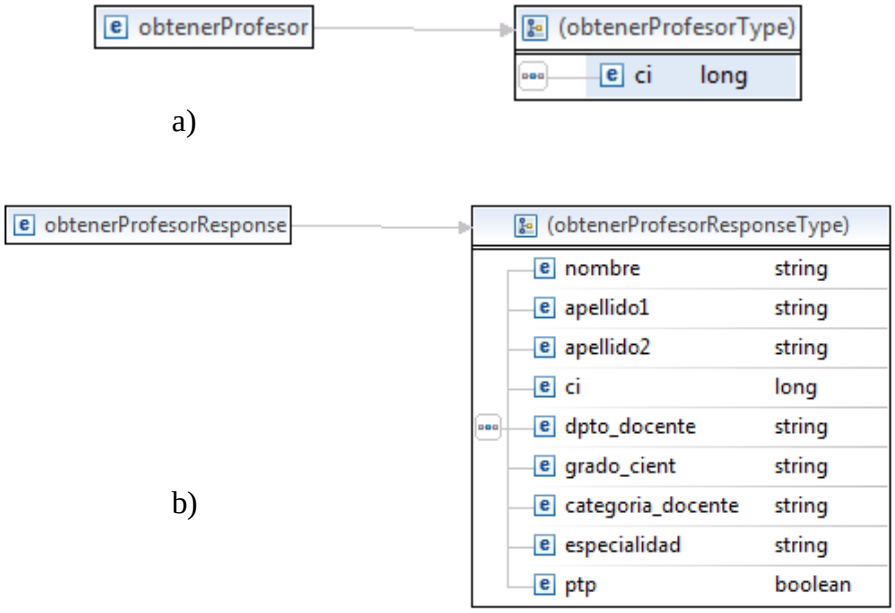


Figura 2.6. Operación *obtenerProfesor* correspondiente a *serviceTrabajador*: a) parámetros de entrada, b) parámetros de salida

Un fragmento del documento WDSL correspondiente al servicio *serviceTrabajador* se muestra en el Anexo 2.

El servicio *serviceAsignatura* permitirá realizar las operaciones relacionadas con la inserción, recuperación, actualización y eliminación de los datos de asignaturas en la base de datos de SAAC y SCCDC. En la figura 2.7 muestra el diseño de este servicio con sus operaciones.

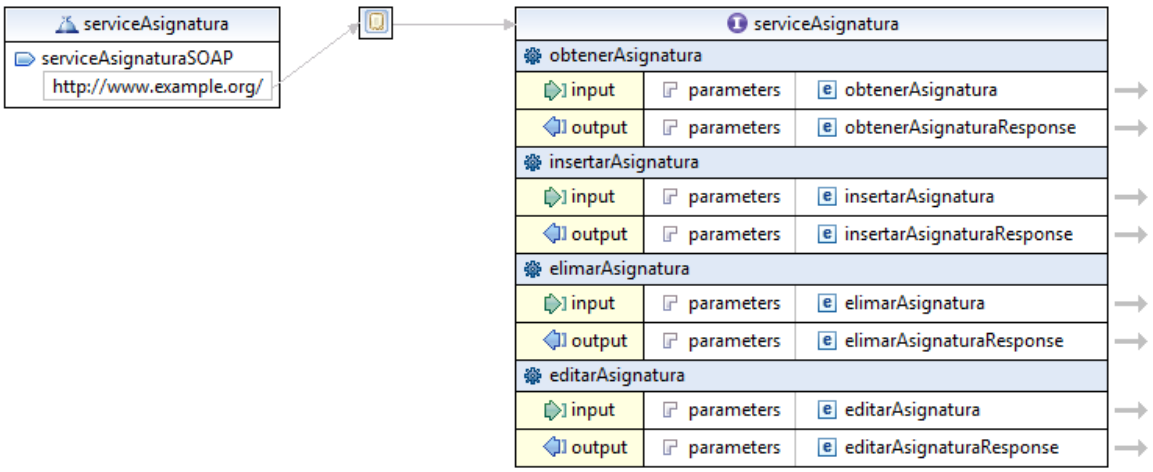
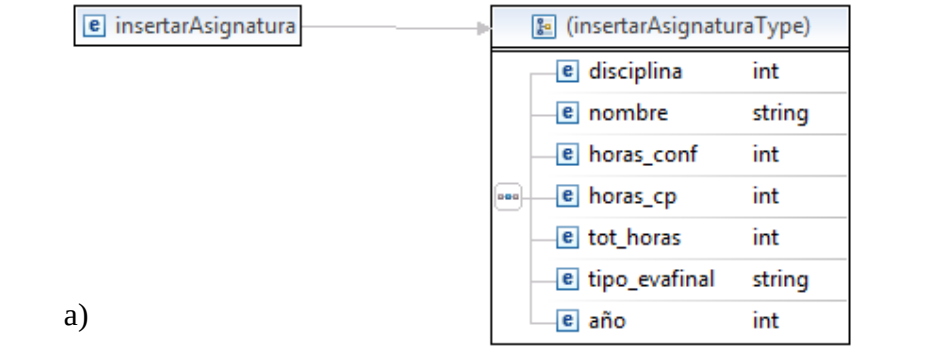


Figura 2.7. Representación de *serviceAsignatura*.

Una de las operaciones de este servicio, *insertarAsignatura* es la encargada de insertar los datos de las asignaturas. La figura 2.8 presenta esta operación y sus parámetros.



a)

b)

Figura 2.8. Operación *insertarAsignatura* correspondiente a *serviceAsignatura*: a) parámetros de entrada, b) parámetros de salida.

El servicio *carreraService* realiza la operación de recuperar los datos relacionados con las carreras en la base de datos de SAAC y SCCDC. La figura 2.9 presenta el diseño de este servicio.

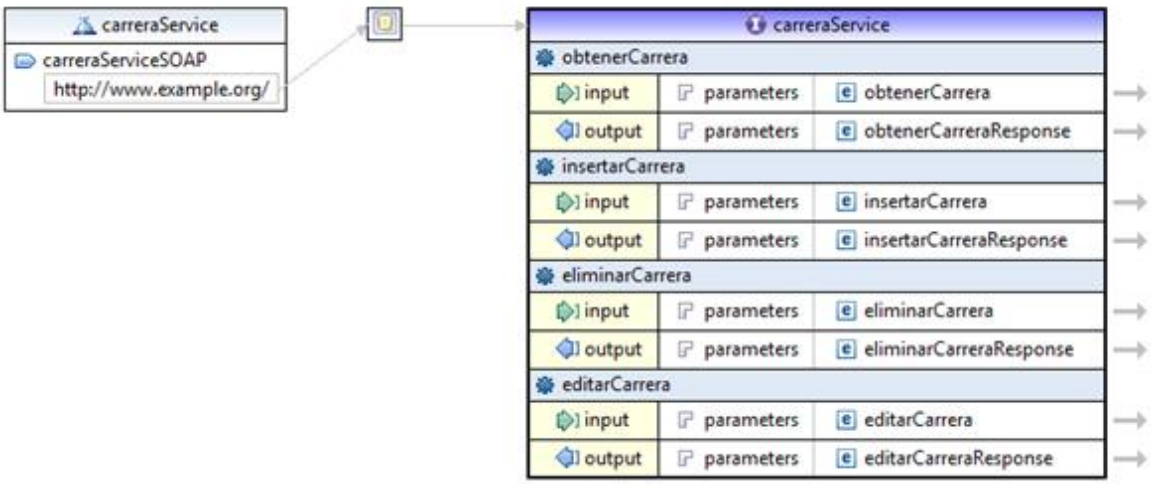


Figura 2.9. Representación de *carreraService*.

La operación *obtenerCarrera* es la encargada de recuperar los datos de las carreras a partir del parámetro de entrada que refiere al identificador de carrera. La figura 2.10 muestra los parámetros de esta operación.

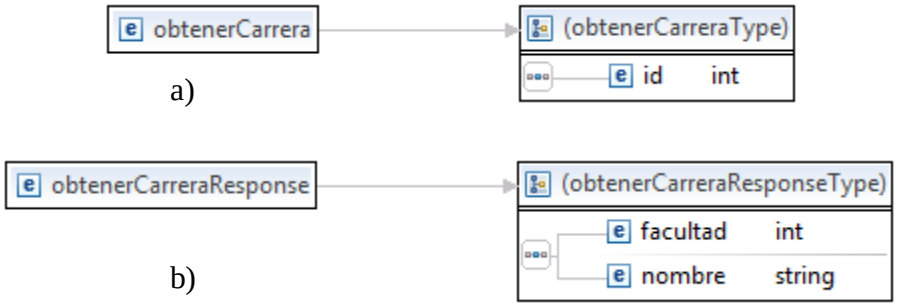


Figura 2.10. Operación *obtenerCarrera* correspondiente a *carreraService*: a) parámetros de entrada, b) parámetros de salida.

La figura 2.11 presenta el servicio *updateService*, de vital importancia en el aseguramiento de la integridad y sincronización de los identificadores entre los sistemas.

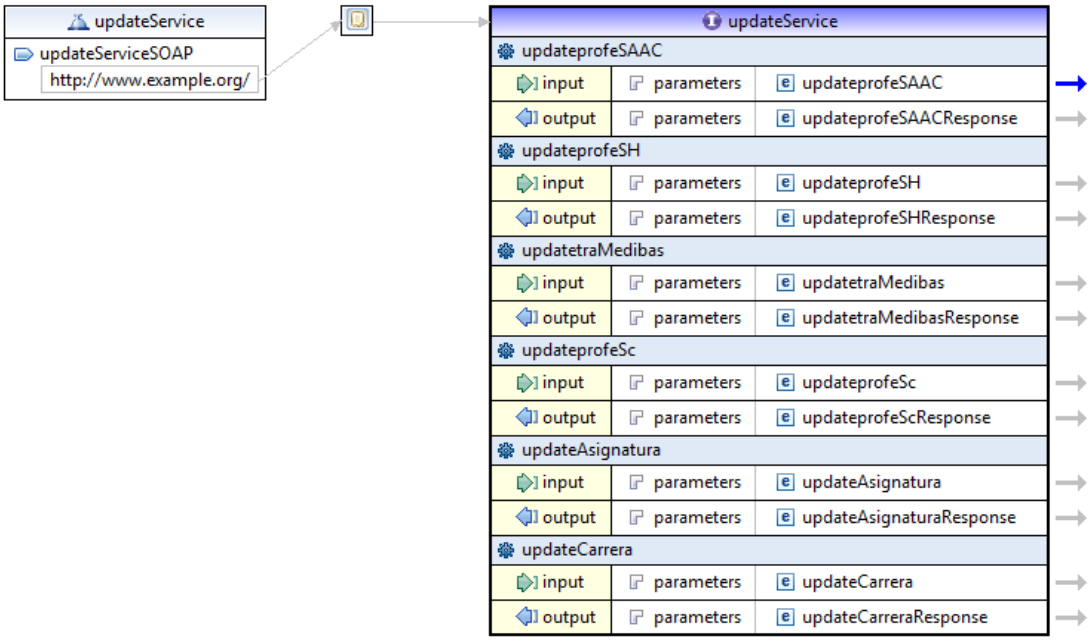


Figura 2.11. Representación de *updateService*.

Este servicio provee operaciones que permitirán garantizar que se actualicen los identificadores de los diferentes sistemas posterior a operaciones como la inserción o eliminación de datos

referentes a trabajadores, en la base de datos TRABAJADORES o datos de asignaturas y carreras en SAAC y SCCDC.

2.2.2 Seguridad en SOA

Para garantizar la seguridad en la arquitectura SOA se propone su aplicación a nivel de mensajes, mediante la utilización del estándar de seguridad *WS-Security*, de manera que nadie pueda leer o modificar los mensajes que son intercambiados entre los servicios.

Generalmente el intercambio de mensajes entre los servicios debe estar asegurado para garantizar la integridad de los datos y la información que se intercambia. Además de los mecanismos de Autenticación y Autorización para el control del acceso a los servicios, una buena práctica es proveer directamente seguridad a los mensajes.

El estándar *WS-Security* define como aplicar diferentes técnicas de seguridad para la autorización, integridad y privacidad con los servicios Web, específicamente en el mensaje SOAP. Describe una manera estandarizada de insertar información de seguridad en forma de *tokens*, codificación, firmas de seguridad, entre otras normas, en el encabezamiento del mensaje SOAP (Josuttis, 2007).

Adicionalmente, la seguridad será también mantenida a través del bus de servicios. Esto se debe principalmente a que si se aplica la seguridad a un alto nivel, como al consumidor de servicio, se tiene la desventaja de que otros consumidores no estarán autenticados o autorizados automáticamente. Esto podría ser un riesgo si los servicios son de fácil acceso y conocidos. Aplicando la seguridad a un nivel bajo, a un servicio específico, implica un mayor trabajo y es más complicado, lo que representa una desventaja. Un buen punto de partida es aplicar la seguridad al ESB, asegurando que sólo las peticiones desde el ESB estén permitidas para los servicios, y no directamente desde los consumidores de servicios.

El uso del ESB garantiza mecanismos para controlar el acceso a los servicios y la seguridad en el intercambio de la información y los mensajes a través del estándar *WS-Security*.

2.3 Herramientas para el diseño e implementación de SOA.

En el mercado mundial, muchas empresas privadas ofrecen herramientas para sustentar un escenario tecnológico que permita obtener los beneficios de la orientación a servicios, pero el precio de las mismas es muy elevado, lo cual representa un inconveniente para Cuba porque

además, afecta el objetivo de lograr una soberanía tecnológica. Se hace necesario realizar un compromiso entre las características de las herramientas a utilizar y las posibilidades reales de su aplicación en el entorno real cubano (Febles, 2012).

A continuación se presentan las principales características de dos herramientas, WSO2 (Web Services Oxygen) y el framework Eclipse, a partir de las cuales fueron creados los contratos de los servicios Web y que se proponen sean empleadas en la implementación de los servicios Web, así como de la infraestructura de la Arquitectura Orientada a Servicios.

2.3.1 WSO2

La suite WSO2 (http://wso2.com) posee las herramientas necesarias para soportar el funcionamiento de SOA. WSO2 es una compañía que ha desarrollado un conjunto de herramientas de código abierto basado en estándares que permite la adopción tecnológica de una Arquitectura Orientada a Servicios. Su plataforma es basada en componentes usando la tecnología OSGI (Open Services Gateway Initiative). Ha sido ubicado como visionario en el Cuadro de Gartner entre las suites de SOA.

Entre las razones de mayor peso para su elección se encuentra su nivel de cumplimiento con los estándares WS-* para el desarrollo de servicios Web, su tipo de licencia (Apache v2), la calidad de su documentación, su característica de multiplataforma, sus facilidades de trabajo en entornos que necesitan de una alta disponibilidad y su buen rendimiento (Febles, 2012).

En una extensa comparativa de las alternativas de código abierto para plataformas SOA en aspectos como la capacidad de integración, el comportamiento en un entorno productivo, la asequibilidad de los productos, los adaptadores disponibles y otras características, la suite **WSO2** sale favorecida frente otras soluciones código abierto existentes (Pozo, 2013).

WSO2 ofrece un amplio conjunto de herramientas, como se presentan en el anexo 3, que contribuyen a implementar una plataforma SOA. Entre los principales productos disponibles, pueden destacarse:

- **WSAS**: Es un servidor que permite almacenar servicios desarrollados usando varios *frameworks* de desarrollo como AXIS2, JAX-WS, o Spring. Además permite adicionarle seguridad de distintos niveles o protegerlos restringiendo su acceso mediante un filtrado de IPs o de dominios.

- **ESB**: Es una implementación del concepto de bus de servicios que permite desarrollar la transparencia de ubicaciones de los servicios que se exponen, y además la transformación y ruteo de mensajes, junto con el monitoreo de las comunicaciones y diferentes mecanismos de seguridad.
- **Registro/Repositorios**: Conocido como GREG (Governance Registry), es una herramienta para gestionar los metadatos de los servicios como pueden ser su WSDL, XSD, políticas, y especificaciones técnicas de los servicios.
- **Servidor de servicios de datos**: Destinada a la creación de servicios de datos, los cuales pueden extraer su información de una gran variedad de fuentes y combinaciones con una alta calidad y en breve tiempo, exponiéndolos como servicios y posibilitando la seguridad necesaria para su consumo.
- **BAM:** Permite monitorizar la actividad de negocio.
- **Identity Server (Servidor de Identidad):** Se puede considerar como una herramienta para la administración y gestión de la seguridad de los servicios. WSO2 provee además otras herramientas necesarias para implementar otros aspectos referentes a la seguridad de la arquitectura SOA.
- **Developer Studio:** Es un framework de desarrollo, basado en Eclipse, para crear y desplegar los distintos artefactos en la plataforma SOA (WSO2 Developer Studio).

2.3.2 Eclipse

Eclipse es un entorno de desarrollo integrado, de código abierto y multiplataforma. Es una potente y completa plataforma de programación, desarrollo y compilación de elementos tan variados como sitios Web, programas en C++ o aplicaciones Java. Si bien las funciones de Eclipse son más bien de carácter general, las características del programa se pueden ampliar y mejorar mediante el uso de módulos. Desde sus orígenes fue concebida para convertirse en una plataforma de integración de herramientas de desarrollo y no se encuentra definida para un lenguaje específico, sino que es un IDE genérico, aunque posee mucha popularidad entre la comunidad de desarrolladores del lenguaje Java**.**

Para el desarrollo de servicios Web, Eclipse provee una serie de funcionalidades para la creación y manipulación de estos servicios. En este sentido, brinda facilidades para el desarrollo de los

servicios Web mediante las dos principales técnicas, diseñando primeramente los contratos (Contract first) o a partir de la implementación (Code first).También permite crear el contrato de un servicio de una manera gráfica y su validación contra el perfil de WS-I, un estándar para la interoperabilidad de servicios entre diversas plataformas de desarrollo.

2.4 Conclusiones del capítulo.

La adopción de una Arquitectura Orientada a Servicios constituye un paso importante con vistas al mejoramiento de la calidad de la distribución de los sistemas del CEI y la facultad MFC. El análisis efectuado como parte del desarrollo de la arquitectura condujo a la identificación de un conjunto de servicios, que junto con una serie de modificaciones propuestas a los sistemas estudiados posibilitará la eliminación de los problemas de duplicidad de datos y de funcionalidades. Como parte del diseño de los servicios se desarrollaron los contratos o documentos WSDL, empleando como apoyo las herramientas de WSO2 y el entorno de desarrollo integrado Eclipse.

CAPÍTULO 3. Interoperabilidad entre el sitio Web del CEI y DSpace.

En este capítulo se expone la solución propuesta para lograr la interoperabilidad entre el sitio Web del CEI y el repositorio DSpace, con lo cual se podrá acceder a la información que mantiene el repositorio. Se propone el uso del protocolo OAI-PMH, detallando los aspectos necesarios para su implementación y posterior empleo desde el sitio Web del CEI.

3.1 Implementación OAI-PMH

La arquitectura de OAI-PMH, como se ha señalado anteriormente, se basa en clientes y servidores. Los primeros son los archivos o repositorios que proporcionan la información, y los segundos son los recolectores o servicios que toman los datos, con el objetivo de incorporarles algún valor añadido y presentarlos a los usuarios finales.

El protocolo OAI-PMH proporciona flexibilidad en su implementación según los objetivos y necesidades de su empleo. Para establecer la interoperabilidad entre el repositorio DSpace y el sitio Web del CEI se adoptó como estructura organizativa la implementación más común del protocolo, al adecuarse a las condiciones de la problemática. De esta forma, se mantendrá un proveedor de servicios que recolecta los metadatos del repositorio DSpace a partir del proveedor de datos y que suministrará la información necesaria al sitio web del CEI.

3.1.1 Proveedor de datos

La arquitectura del proveedor de datos (PD) contempla un conjunto de componentes entre los que se incluyen:

- **Analizador Sintáctico de Argumentos.** Que permite validar las peticiones OAI que son enviadas desde el proveedor de servicios.
- **Generador de Errores.** Que posibilita crear respuestas XML con los mensajes de error codificados.
- **Consulta a Base de Datos / Extracción de Metadatos Locales**. A partir del cual se recuperan los metadatos del repositorio accediendo a la base de datos, en correspondencia con el formato de metadatos requerido.
- **Generador XML / Creación de la Respuesta.** Encargado de crear las respuestas XML con la información de los metadatos codificada.

El funcionamiento del proveedor de datos se representa a través del diagrama de flujo de la figura 3.1, el cual se explica a continuación (Carpenter, 2003):

El diagrama de flujo muestra el procesamiento de una petición al Proveedor de Datos. El Proveedor de Datos al recibir una petición OAI tiene que analizar la consulta y decidir si esta se encuentra dentro de los seis tipos válidos de petición. En caso de no ser válida la petición (el parámetro *Verb* posee un valor no estándar) se traduce en un mensaje de error al Proveedor de Servicios (error *badVerb*).

Cuando el tipo de petición suministrado, como en el ejemplo que se propone, es *ListIdentifers* el analizador sintáctico tiene que comprobar el parámetro *metadataPrefix*, pues este argumento es obligatorio para la petición *ListIdentifiers*. Si no se ha proporcionado este parámetro la única posibilidad de que la petición sea válida es que contenga el parámetro *resumptionToken* que debe ser conocido por el Proveedor de Datos y que identifica la reanudación de una petición *ListIdentifiers* previa que emitió un resultado incompleto. En este caso, el Proveedor de Datos lee los parámetros, almacenados localmente, que representan los argumentos de la petición original y la información que indica cuantos identificadores han sido proporcionados ya al Proveedor de Servicios. Si el argumento *resumptionToken* está vacío (error *badArgument*) o tiene un valor desconocido (error *badResumptionToken*) por el Proveedor de Datos se tienen que generar entonces un mensaje de error.

El único valor válido del parámetro *metadataPrefix* es oai_dc porque el Proveedor de Datos del repositorio DSpace sólo ofrece *Sets* de metadatos en el esquema Dublin Core no cualificado. Verificados estos aspectos, los restantes parámetros opcionales, que en el diagrama de flujo están descritos de manera informal en aras de la simplicidad, tienen que ser también validados por el analizador sintáctico. En este proceso, se deben proporcionar mensajes de error si los parámetros tienen valores ilegales o si la consulta contiene otros parámetros no permitidos para este tipo de petición.

Posteriormente, los parámetros recibidos por la consulta tienen que ser ensamblados en una consulta SQL que luego se debe enviar a la base de datos. Si esto produce más de 100 registros, el Proveedor de Datos tiene que generar una nueva señal de reanudación y almacenarla localmente, junto con los parámetros de la consulta. La señal de reanudación ha de incluirse

también en la respuesta XML al Proveedor de Servicios junto con los identificadores devueltos por la base de datos.

El procesamiento de las restantes peticiones es similar al antes planteado y sigue el mismo esquema básico: verificar los parámetros, generar la consulta SQL y formar el documento de respuesta XML con los resultados obtenidos.

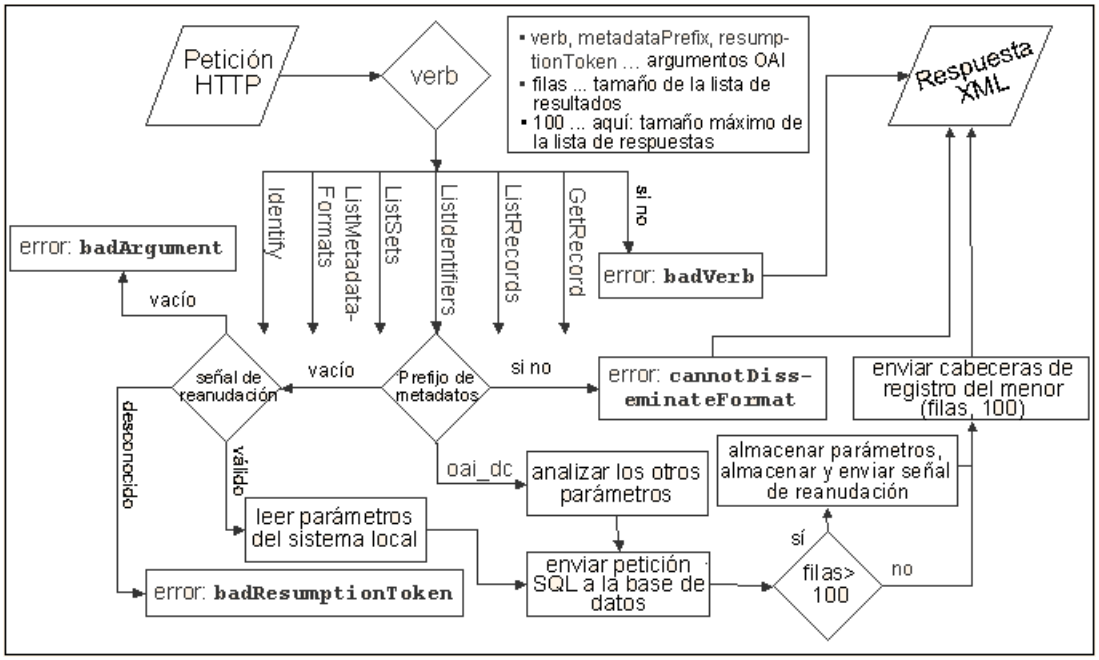


Figura 3.1. Funcionamiento Proveedor de Datos.

La plataforma DSpace del CEI en su versión 1.4 implementa la versión 2.0 del protocolo OAI-PMH como proveedor de datos.

DSpace construye un archivo ([dspace-source]/build/dspace-oai.war) para recibir y responder las peticiones OAI-PMH a través de HTTP. De esta forma, en la configuración de DSpace, la URL base mediante la cual se procesan las peticiones OAI-PMH corresponde con: *http://localhost/dspace-oai/request*.

Como se ha explicado, OAI-PMH permite a los repositorios organizar sus artículos en una jerarquía de *Sets*.

DSpace expone sus *Colecciones* como *Sets*. Cada colección posee su correspondiente *Set* OAI, que pueden ser descubiertos por el recolector a través del verbo *ListSets*. De esta forma, el nombre de la colección es también el nombre del correspondiente *Set*.

Una característica importante del proveedor de datos OAI es el control de flujo. Si el proveedor de datos recibe una petición para muchos datos, este puede enviar parte de los datos con una señal de reanudación. El recolector puede luego con la señal de reanudación recuperar el resto de los datos.

DSpace mantiene la señal de reanudación para las peticiones *ListRecords* de OAI-PMH. Cada petición *ListRecords* no retornará más de 100 artículos. Este límite es fijado como parte de la configuración de DSpace y es colocado en org.dspace.app.oai.DSpaceOAICatalog.java (MAX_RECORDS).

3.1.2 Proveedor de servicios

Los componentes necesarios que conforman la arquitectura de un proveedor de servicios (PS) incluyen (Carpenter, 2003):

- **Gestor de archivos.** Que permite la selección de los repositorios a recolectar.
- **Componente de peticiones.** El cual crea las peticiones HTTP y las envía al repositorio OAI (Proveedores de Datos). Los metadatos se solicitan empleando los verbos permitidos por OAI-PMH. Posibilita la recolección selectiva utilizando parámetro *Set*.
- **Planificador de recolección.** Lleva a cabo la recuperación regular y sincronizada de los archivos o repositorios asociados.
- **Control de flujo.** Que se implementa a través de la señal de reanudación, con el fin de obtener la lista de resultados completa de una petición. Para proporcionar control de flujo, la señal de reanudación debe "reconocer" que la respuesta del Proveedor de Datos contiene una lista incompleta de resultados y reenviar entonces la petición OAI al Proveedor de Datos para obtener la siguiente parte del listado.
- **Mecanismo de actualización.** Que ejecuta la consolidación de los metadatos que han sido recolectados con anterioridad (agrupación de los metadatos antiguos y los nuevos).
- **Analizador sintáctico de XML.** Que examina las respuestas recibidas de los repositorios, las valida utilizando el esquema XML, y convierte los metadatos codificados en XML a la estructura de datos interna.

- **Normalizador.** Que permite transformar los datos que se encuentran en diferentes formatos de metadatos en una estructura homogénea. Puede establecer equivalencias o traducir entre diferentes lenguajes.
- **La base de datos.** Recibe el resultado de la equivalencia, realizada por el normalizador, entre la estructura XML de los metadatos y la base de datos relacional que gestionará los valores de los elementos.
- **El detector de duplicados.** Fusiona registros idénticos de diferentes Proveedores de Datos.
- **Módulo de servicios.** El cuál proporciona el servicio real al usuario. El fundamento de los servicios que se brindan es la recolección y almacenamiento de los registros de los archivos asociados.

La figura 3.2 presenta la arquitectura de un proveedor de servicios y la relación entre sus componentes.

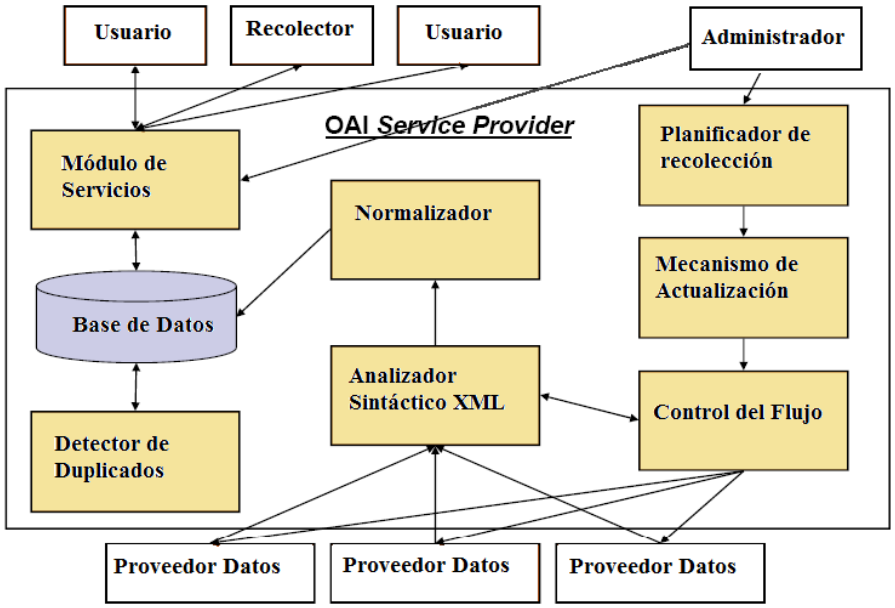


Figura 3.2. Arquitectura de Proveedor de Servicios.

Para la implementación del proveedor de servicio se propone el empleo de la herramienta Open Archives Harvester un sistema de código abierto para la recolección de metadatos desarrollado por Public Knowledge Project (PKP).

Open Archives Harvester está implementado en el lenguaje orientado a objetos PHP (http://www.php.net) empleando el sistema de plantillas Smarty (http://smarty.php.net) para la abstracción de interface de usuario. Los datos se almacenan en una base de datos SQL, con llamadas a la base de datos abstraídas a través de la librería ADODB Database Abstraction (http://adodb.sourceforge.net) (Smecher et al., 2012b).

Los requerimientos tecnológicos de esta herramienta incluyen:

- Soporte PHP (4.2.x o superior)
- Gestor de Base de Datos MySQL (3.x o superior)
- Servidor Apache (1.3.2x o superior) o Apache 2.0 (2.0.4x o superior). Puede ser empleado también Microsoft IIS 6 (este requiere PHP 5.x)
- Sistema Operativo: Linux, BSD, Solaris, Mac OS X, Windows

Open Archives Harvester está diseñado para ser una herramienta flexible en la recuperación, almacenado, indexación, búsqueda de datos desde una variedad de diferentes tipos de fuentes. Varias partes de este proceso es abstraído usando plugins para permitir futuras extensiones; por ejemplo los esquemas de metadatos están cada uno implementados como plugins y otros pueden ser incorporados colocando un nuevo plugin en el directorio apropiado. De igual forma, los protocolos de recolección de metadatos como OAI-PMH, son implementados como plugins (Smecher et al., 2012b). Las características principales, por tanto, de esta herramienta incluyen:

- Soporte incorporado para OAI Protocol for Metadata Harvesting, en las versiones 1.1 y 2.0.
- Soporte incorporado para los formatos de metadatos: Dublin Core, MODS, and MARC.
- Soporte adicional para que otros protocolos de recolección y formatos de metadatos puedan ser instalados vía plugins.

Como en una estructura Modelo-Vista-Controlador, el diseño de esta herramienta separa en diferentes capas el almacenamiento y representación de los datos, la presentación de las interfaces de usuario y el control. De esta forma los componentes más representativos de su arquitectura incluyen (Smecher et al., 2012b):

- **Smarty templates**, responsable de reunir las páginas HTML que se visualizarán a los usuarios.

- **Page classes**, recibe las peticiones desde los usuarios (navegador web), delegando cualquier procesamiento requerido a varias otras clases e invocando al apropiado Smarty templates para generar una respuesta.
- **Model classes**, que implementa los objetos PHP representando varias entidades del sistema, como los repositorios.
- **Data Access Objects**(DAOs), generalmente proporcionan funciones de actualización, eliminación y creación para sus asociados model classes, es responsable por toda la interacción con la base de datos.
- **Support classes**, proporciona funcionalidades centrales, clases y funciones comunes heterogéneas, entre otros.

Instalado el sistema, es necesario autentificarse como administrador para acceder a las funciones de administración. Los elementos más importantes del administrador abarcan la selección de los archivos o repositorios que serán objetos de la recolección de sus metadatos, proceso este que depende del tipo de recolector empleado. En el caso del recolector OAI, datos como la URL del repositorio, la URL base OAI, el esquema de metadatos a emplear, entre otros, deben ser proporcionados.

La figura 3.3 muestra la página que permite la recuperación regular de los metadatos del repositorio especificado a partir del recolector OAI, función esta del administrador. La herramienta permite la recolección selectiva a partir de los *Sets* y las fechas de los registros. Teniendo en cuenta el alcance de la información que debe ser visualizada en el sitio web del CEI será necesario una recolección selectiva empleando los *Sets* que agrupan a las publicaciones científicas por año.

Los metadatos recolectados son almacenados en una base de datos SQL empleada por el sistema, que será utilizada posteriormente para realizar búsquedas locales.

Se han desarrollado varias versiones de esta herramienta. Específicamente, la versión 2.3.2, última desarrollada hasta el momento, fue examinada para los propósitos de esta tesis, logrando recuperar los datos resguardados en el repositorio DSpace, de acuerdo a las especificaciones requeridas.

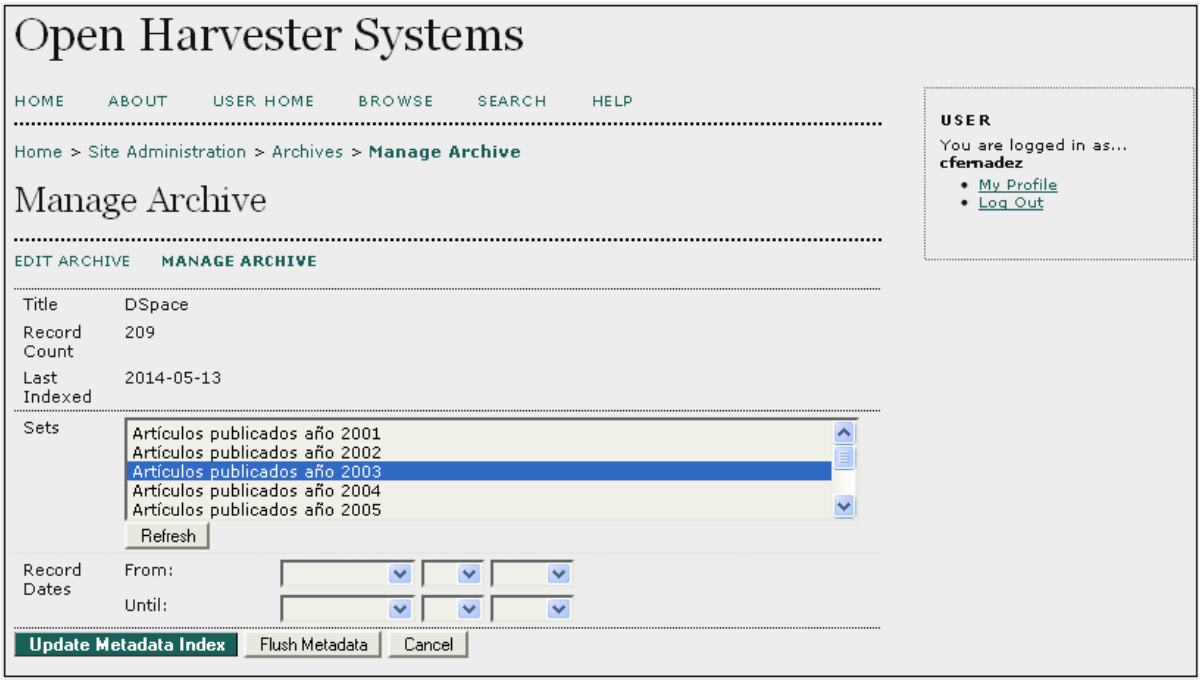


Figura 3.3. Herramienta OHS.

Para profundizar en aspectos técnicos de la herramienta se puede consultar (Smecher et al., 2012b), donde se presentan muestras de código, estructura de archivos, referencia a las clases, diseño de la base de datos, seguridad, entre otros detalles propios de la herramienta. Para una descripción detallada del proceso de instalación y principales funcionalidades de la herramienta consultar (Smecher et al., 2012a).

3.2 Adaptación del PS en el Sitio Web del CEI

El sitio Web del CEI ofrece información vinculada al trabajo que se realiza en el centro, esto incluye información sobre su misión, objetivos, colectivo laboral, grupos investigativos, entre otros. Entre la información que debe ofrecer sobre los profesores e investigadores se encuentra el currículo vitae de los mismos, que contiene entre otros datos, la producción científica. Este servicio, aunque se encuentra referenciado desde las páginas de sitio, aún no se ha implementado debido a que, como ya se ha mencionado, para la conformación de la producción científica es necesario acceder y recuperar la información que se encuentra resguardada en el repositorio DSpace. La figura 3.4 muestra la página de *Profesores* del sitio Web del CEI donde se hace referencia en los datos del profesor a *currículo*.

Figura 3.4. Sito Web del CEI.

La conformación de la producción científica de un profesor empleando la información que ya se encuentra almacenada en el repositorio DSpace resulta necesaria, pues de otra forma al no existir comunicación entre los sistemas se estaría provocando duplicación de información, con los sucesivos problemas, analizados previamente en esta tesis, que esta característica puede provocar. A lo anterior se agrega el hecho de que el repositorio DSpace es el sitio oficial donde se deposita, por parte de la directiva del centro, el balance científico que cada año se desarrolla, con lo cual puede asegurarse la veracidad de la información que se encuentra registrada en el repositorio.

Como se planteó en el epígrafe 3.1 el proveedor de servicio es el encargado de recuperar la información desde el repositorio DSpace enviando peticiones que son atendidas por el proveedor de datos. Para desarrollar las funciones del proveedor de servicio se propuso el empleo de la herramienta Open Archives Harvester.

Las características de esta herramienta, principalmente ser un sistema de código abierto, posibilita el estudio, análisis, comprensión y modificación de sus clases y funciones principales.

Partiendo de esta base se propone emplearlas funciones de la herramienta que permiten obtener y visualizar las publicaciones científicas de un profesor, así como los principales metadatos que describen el documento, con el objetivo de exponer estos resultados desde el ambiente del sitio web del CEI, sin modificar con esto la interfaz que ofrece la herramienta para facilitar la interacción con los usuarios.

Para llevar a cabo lo antes expuesto fue necesario, primeramente, localizar y analizar las clases y funciones que, como parte de la implementación de la herramienta, se encargan de las acciones antes descritas. Las clases y subclases involucradas en estas funciones se detallan a continuación:

- La clase ***Handler*** recoge varias subclases, en particular ***MysqlIndexSearchHandler*** implementa la función ***results($args)*** que realiza la búsqueda de los datos a partir del argumento que recibe e invoca el código necesario encargado de mostrar los resultados obtenidos desde la página html. La implementación de la clase se encuentra en *[dir_ohs]/plugins/generic/mysqlIndex/MysqlIndexSearchHandler.inc.php*.

- La clase ***Record*** que describe las propiedades básicas de los elementos recolectados, implementa la función ***displaySummary()*** que identifica el esquema de metadatos que describe la información que se solicita e invoca a la función ***displayRecordSummary()***. Esta clase se implementa en *[dir_ohs]/classes/record/Record.inc.php*.

- La función ***displayRecordSummary()***, que muestra los resultados obtenidos en la página HTML, es implementada en la clase ***SchemaPlugin***,la cual como indica su nombre maneja los esquemas de metadatos. Esta clase es implementada en *[dir_ohs]/classes/plugins/SchemaPlugin.inc.php*.

- La clase ***RecordHandler*** implementa la función ***view($args)*** que obtiene los metadatos de un elemento especificado. En este proceso invoca a la función ***setupTemplate()*** que permite la visualización de los metadatos desde la página html. La implementación de esta clase se encuentra en: *[dir_ohs]/pages/record/RecordHandle.inc.php*.

Con el propósito de no modificar la interfaz que brinda la propia herramienta, se realizaron modificaciones al código que permitieron mostrar los resultados de la búsqueda desde una interfaz propia. Se introdujeron nuevas funciones que fundamentalmente realizan las mismas

acciones que las funciones originales, pero que redireccionan los resultados hacia interfaces propias implementadas. De esta forma las nuevas funciones fueron nombradas: *midisplayRecordSummary()*, *midisplaySummary()*, *miresults($args)*, *view($args)*, *setupTemplate()*.

Básicamente, a través de la URL se realiza una petición a Open Archives Harvester. La URL empleada debe ser *[dir_ohs]/index.php?page=misearch&op=miresults&Query=[autor]*, donde a partir del nombre del autor (profesor) se realizará una búsqueda que permitirá identificar sus publicaciones científicas previamente recolectadas. La figura 3.5 muestra los resultados obtenidos tras realizar esta petición desde una página ejemplo, que representa el sitio web del CEI.

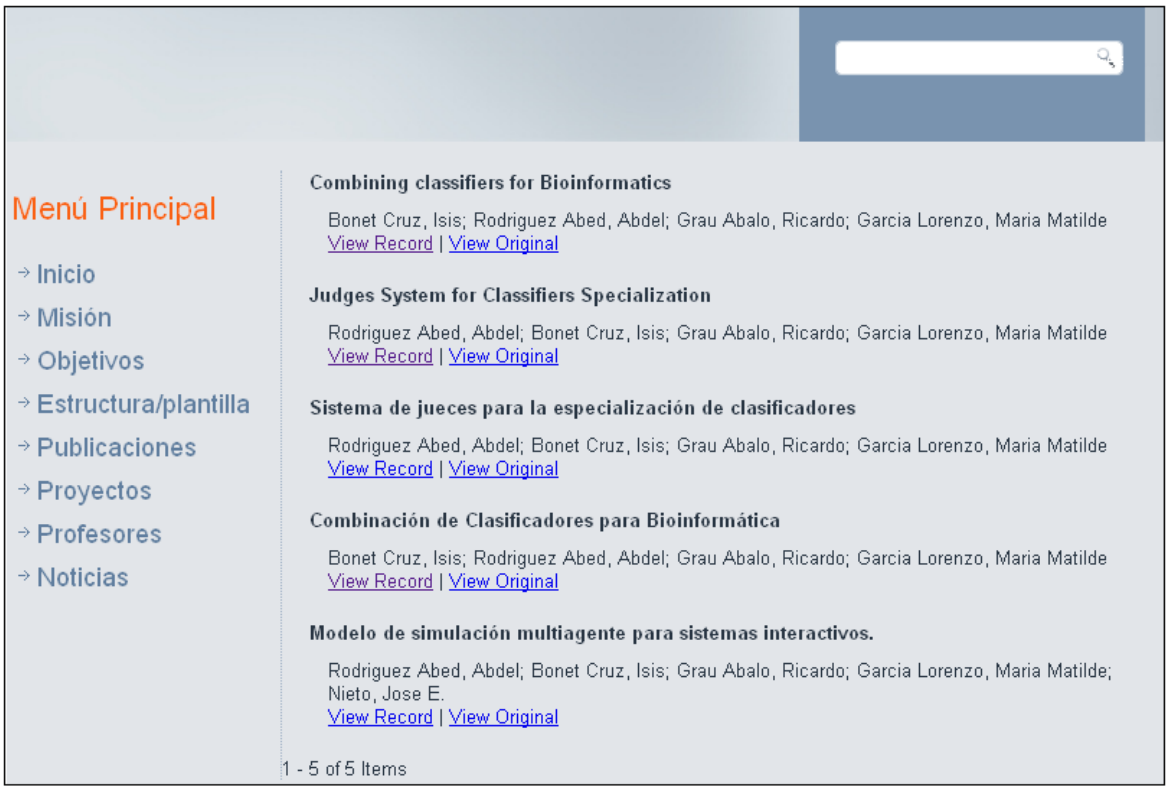


Figura 3.5. Metadatos recolectados del repositorio DSpace.

Junto con los datos de todas las publicaciones del autor seleccionado se muestran dos enlaces. El enlace *View Record* (*[dir_ohs]/index.php/page=record&op=miview&path[]=[valor]*), donde el valor es un número asignado internamente por el sistema para su identificación, permite visualizar los principales metadatos que describen esta publicación, previamente recolectados desde el repositorio DSpace. Por su parte, el enlace *View Original* brinda la posibilidad al

usuario de vincularse directamente con la interfaz del repositorio DSpace donde junto con los metadatos se podrá consultar además el documento original.

Como parte de los metadatos que el repositorio DSpace emplea en la descripción de un material almacenado digitalmente se encuentra el URI que significa "Identificador Único de Recurso" y que DSpace incorpora como forma de acceso a sus recursos desde el exterior. Por defecto a este metadato se le asigna una URL perteneciente al sistema *Handle*. Ejemplo de esta URL es: *http://hdl.handle.net/123456789/123*.

El Sistema *Handle* permite la asignación de identificadores persistentes a los recursos de información u objetos digitales existentes en Internet, estructurándose en dos partes:

- **Prefijo**: Identifica al productor del identificador
- **Sufijo**: Identifica a cada uno de los documentos o recursos digitales

Los identificadores persistentes surgen para solucionar el problema de los cambios de ubicación, dominio y/o nombre de los archivos en Internet. Su objetivo es redireccionar o permitir el acceso a los documentos, aunque estos hayan cambiado de ubicación en la red.

Para el empleo de estos identificadores es necesario adquirir una licencia *handle* previo pago, lo cual imposibilita emplear esta facilidad como parte de la solución de esta tesis. Teniendo en cuenta lo anterior se propone que desde el repositorio DSpace, posterior a la inserción de un documento u otro objeto digital se edite este metadato y se modifique la URL empleando en su lugar la URL del repositorio, como *http://[dir_dspace]/123456789/123*.

3.3 Conclusiones del capítulo

El empleo del protocolo OAH-PMH proporciona la interoperabilidad necesaria entre el sitio web del CEI y el repositorio DSpace, al facilitar la recuperación regular de los metadatos, que se encuentran almacenados en el repositorio, referentes a la producción científica de los profesores. Como parte de los servicios que debe brindar el sitio web del CEI, los metadatos recuperados serán expuestos desde el sitio, conformando el currículo de los profesores.

La implementación del proveedor de datos y el proveedor de servicios es fundamental para el funcionamiento del protocolo. Particularmente, el repositorio DSpace implementa el proveedor de datos de OAI-PMH como forma de exponer sus metadatos. En la implementación del

proveedor de servicios fue empleado la herramienta de código libre Open Archives Harvester, diseñada para la recuperación, almacenado, indexación y búsqueda de datos desde varias fuentes. Los criterios que condujeron a su empleo incluyen principalmente las facilidades en la recuperación de los metadatos al permitir la recuperación de estos desde varias fuentes, posibilidades de modificación del esquema de metadatos a emplear en la recolección, así como en el protocolo de intercambio de información. Aunque no todas estas funcionalidades son necesarias teniendo en cuenta el alcance de esta tesis, posibilitarían en el futuro ampliar los servicios del sitio web del DSpace y adaptarse ante modificaciones que puedan realizarse al repositorio DSpace.

CONCLUSIONES

1. A partir del estudio de la informatización de la facultad de MFC y CEI se identificaron un conjunto de problemas referentes a la duplicación de datos y funcionalidades, así como de ausencia de interoperabilidad, causas fundamentales de un desarrollo aislado e independiente de los sistemas, en ocasiones desarrollados sobre plataformas diferentes, lo que provoca inconsistencia en la información y en las funcionalidades de los sistemas.
2. El estudio del estado del arte permitió la identificación de varios patrones de diseño que permiten la integración de las tecnologías de la información en una institución. Basados en estándares y protocolos cada uno de estos patrones o arquitecturas posee como objetivo eliminar la estructura de los sistemas de información tradicionales que se presentan como sistemas redundantes e inaccesibles y desarrollar nuevos modelos que introduzcan y mejoren la interoperabilidad entre los sistemas. Estos patrones conllevan ventajas y desventajas con su adopción.
3. El análisis de las arquitecturas de integración referenciadas en la literatura condujo al diseño de una arquitectura orientada a servicios por las ventajas y flexibilidades que ofrece a la solución de las problemáticas identificadas. Como parte de la infraestructura SOA se proponen un conjunto de cambios en las bases de datos de los sistemas involucrados que permitirán eliminar la duplicidad de datos y funcionalidad presente entre los mismos.
4. El análisis y diseño orientado a servicios posibilitó la identificación de un conjunto de servicios que expondrán funcionalidades presentes entre los sistemas estudiados y que permitirán la puesta en práctica de las modificaciones propuestas. Como parte del diseño fueron desarrollados los contratos de servicio, lo que facilita y sienta las bases para su posterior implementación.
5. El acceso a la información de DSpace desde el sitio web del CEI se implementó a partir del empleo del protocolo OAI-PMH. El empleo de este protocolo proporciona la interoperabilidad necesaria entre ambos sitios, al facilitar la recuperación regular de los metadatos que se encuentran almacenados en el repositorio, referente a la producción científica de los profesores. La correcta implementación del proveedor de datos y el proveedor de servicios es fundamental para el funcionamiento del protocolo, esto condujo al empleo de la herramienta de código abierto Open Archives Harvester en la implementación

del proveedor de servicio, al ofrecer esta todas las funcionales que debe poseer el proveedor de servicio OAI.

RECOMENDACIONES

1. Desarrollar la arquitectura orientada a servicios propuesta, implementando las modificaciones y los servicios Web identificados. Realizar la implementación de los servicios web mediante las herramientas referenciadas, la suite WSO2 junto con el Eclipse IDE, utilizando los contratos de servicios (wsdl) desarrollados.
2. Brindar, como parte de la información del sitio web del CEI, la producción científica de los profesores e investigadores del centro, implementado la solución propuesta en esta tesis que permitirá el acceso a la información del repositorio DSpace del CEI donde actualmente se encuentra almacenada esta información.

REFERENCIAS BIBLIOGRÁFICAS

Azán, Y. & Díaz, A. (2009): Una experiencia sobre integración de aplicaciones informáticas. *Última visita:* Abril 2014. *p* 9.

Barrueco, J. M. & Coll, I. S. (2003). Open archives initiative. Protocol for metadata harvesting (OAI-PMH): descripción, funciones y aplicaciones de un protocolo. El profesional de la información, 12(2), *pp* 99-106.

Bueno de la Fuente, G. & Rodríguez, D. (2007): Herramientas de software para OAI-PMH. In *La Iniciativa de Archivos Abiertos (OAI): situación y perspectivas en España y Latinoamérica*. Bogotá. *pp* 247-302.

Carpenter, L. (2003): OAI for Beginners - the Open Archives Forum online tutorial. http://www.oaforum.org/tutorial/english/OpenArchivesForum-OAI-PMHOnlineTutorial.htm. *Retrieved:* Enero, 2014

Castro, O. A. (2013): Sistema para el control de la carga docente en el departamento de Ciencia de la Computación Universidad Central "Marta Abreu" de las Villas. *p* 58.

Davis, J. (2009): Open Source SOA, Manning Publications Co. *p* 449.

Dikmans, L. & Van Luttikhuizen, R. (2012): SOA Made Simple, Packt Publishing Ltd. *p* 292.

Erl, T. (2005): Service-Oriented Architecture: Concepts, Technology, and Design. Edtion ed. Prentice Hall PTR.

Erl, T. (2007): SOA: Principles of Service Design, Edtion ed. Prentice Hall *p* 608.

Erl, T., Karmarkar, A., Walmsley, P., Haas, H., Yalcinalp, U., Canyang, K. L., Orchard, D., Tost, A. & J., P. (2010): Web Service Contract Desing and Versioning for SOA, Prentice Hall.

Febles, O. (2012): MIDAC: Modelo para el desarrollo de aplicaciones compuestas basadas en arquitecturas orientadas a servicios. Tesis presentada en opción al grado científico de Doctor en Ciencias Técnicas. Universidad de las Ciencias Informáticas. La Habana. *p* 131.

González, H. (2013): Sitio Web para publicar y controlar el horario docente de la Facultad MFC. Facultad de Matemática, Física y Computación. Universidad Central "Marta Abreu" de las Villas. *p* 63.

Group, G. (2006): Research Architecture. www.gartner.com/technology/research/methologies/. *Retrieved:* Febrero, 2014

Hohpe, G. (2002): Enterprise Integration Patterns. *Última visita:* Febrero 2014. *p* 36.

Hohpe, G. & Woolf, B. (2004): Enterprise Integration Patterns. Designing, Building, and Deploying Messaging Solutions, Addison-Wesley. *p* 680.

Josuttis, N. (2007): SOA in Practice, August 2007 ed., O'Reilly. *p* 344.

Kanneganti, R. & Chodavarapu, P. (2008): SOA Security, Manning Publications Co. *p* 511.

Lagoze, C. & Van de Sompel, H. (2008): The Open Archives Initiative Protocol for Metadata Harvesting http://www.openarchives.org/OAI/2.0/openarchivesprotocol.htm. *Retrieved:* Febrero, 2014

Manero, B. & Fernández, B. (1990). IEEE Definición de interoperabilidad

Marshall, J. (2012): HTTP Made Really Easy. http://www.jmarshall.com/easy/http/. *Retrieved:* Marzo, 2014

Martínez, G. A. (2009): Implementación de un Repositorio Digital empleando tecnologías de código abierto. Tesis para optar al Título de Ingeniero Civil en Informática. Facultad de Ciencias de la Ingeniería. Universidad Austral de Chile. Valdivia *p* 93.

Medina, G. (2013): Sistema de Información de apoyo a la Autoevaluación de la Carrera Ciencia de la Computación (SAAC). Universidad Central "Marta Abreu" de las Villas. *p* 70.

Microsoft, C. (2006): La Arquitectura Orientada a Servicios (SOA) de Microsoft aplicada al mundo real. *Última visita:* Marzo 2014. *p* 24.

Pozo, F. (2013): WSO2 como plataforma SOA open source. http://blog.gfi.es/wso2-como-plataforma-soa-open-source/. *Retrieved:* Mayo, 2014

Pulier, E. & Taylor, H. (2006): Understanding Enterprise SOA, Manning Publications Co. *p* 281.

Radenakers, T. & Dirksen, J. (2009): Open Source ESBs in Action, Manning Publications Co. *p* 529.

Reyes Del Toro, M. d. P. (2007): DSpace. *Última visita: Febrero p 6*

Schmutz, G., Liebhart, D. & Welkenbach, P. (2010): Service-Oriented Architecture: An Integration Blueprint, Packt Publishing Ltd. *p* 240.

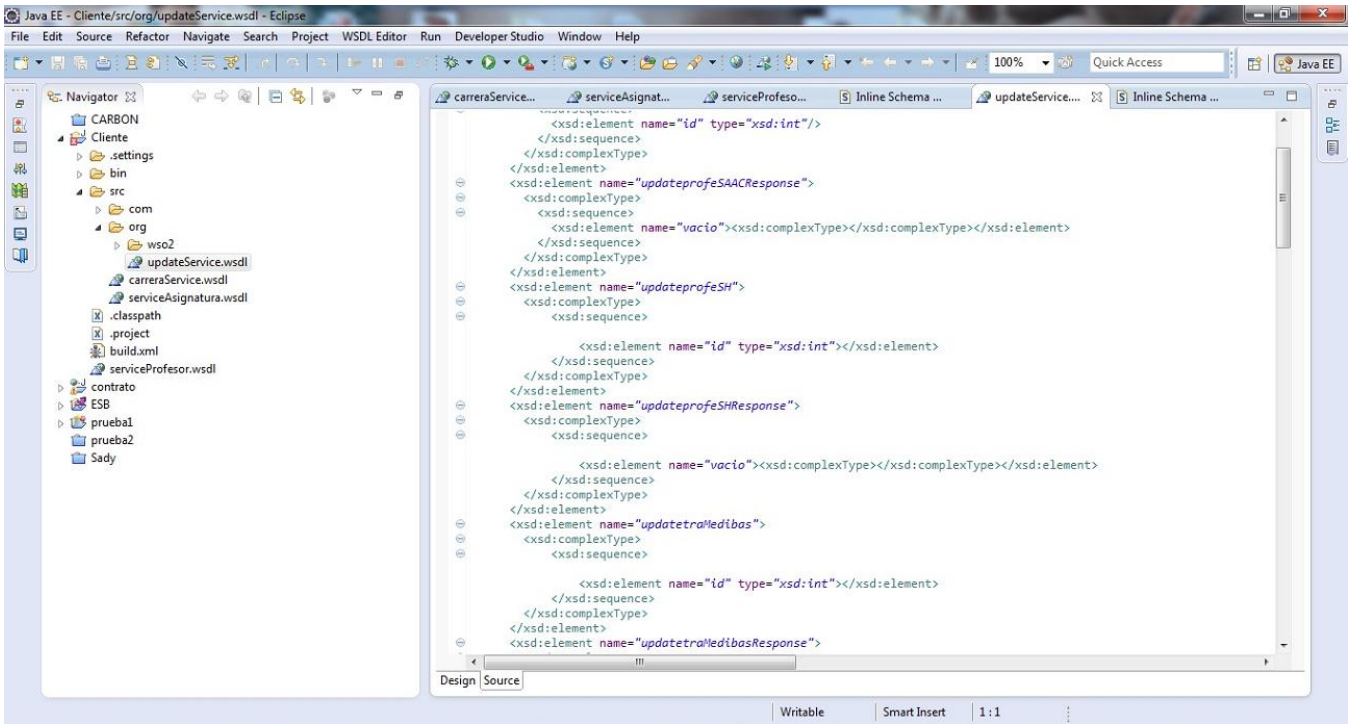
Smecher, A. & Willinsky, J. (2012a): PKP Harvester2 in an Hour. Administrator's Guide. http://pkp.sfu.ca/ohs/ohs_documentation. *Retrieved:* Mayo, 2014

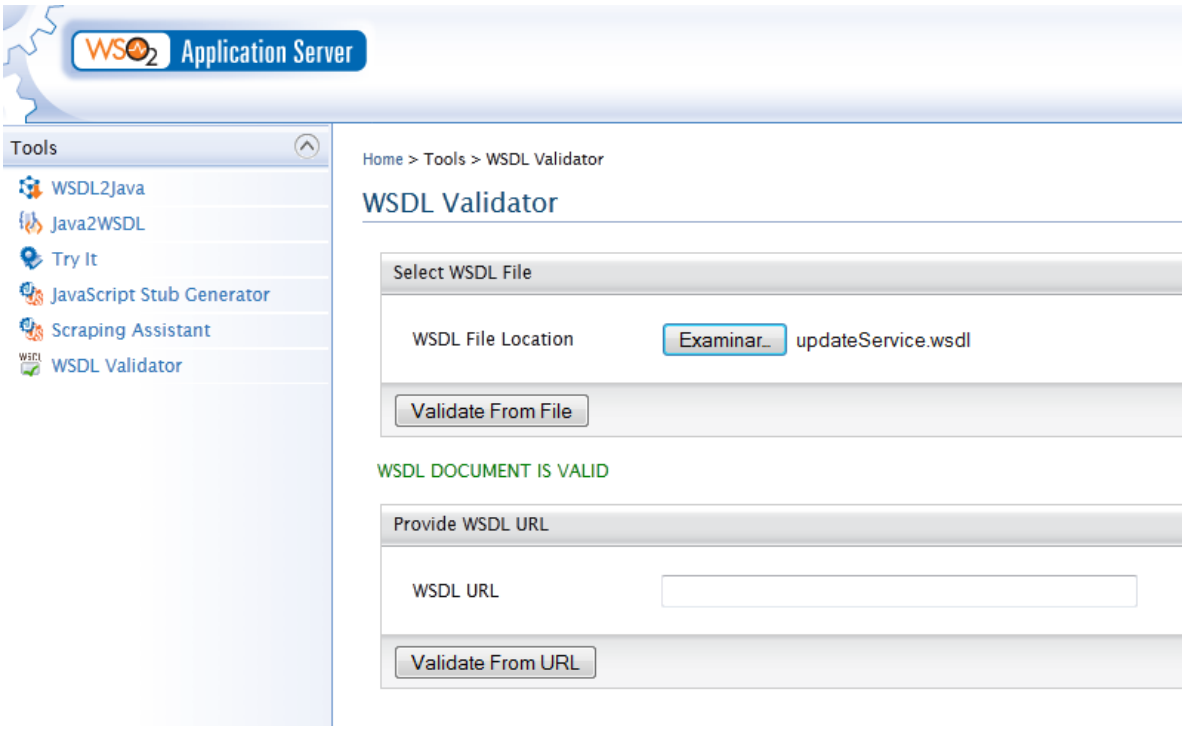
Smecher, A. & Willinsky, J. (2012b): PKP Harvester2. Technical Reference. http://pkp.sfu.ca/ohs/ohs_documentation. *Retrieved:* Mayo, 2014

Tansley, R., Bass, M., Branschofsky, M., Carpenter, G., McClellan, G. & Stuve, D. (2006): DSpace System Documentation. http://sourceforge.net/projects/dspace/files/DSpaceStable/1.4.2/dspace-1.4.2-source.zip/download. *Retrieved:* Febrero, 2014

ANEXOS

ANEXO 1 Diseño y validación del contrato *updateService.wsdl*





ANEXO 2 Fragmento del contrato del servicio *serviceTrabajador*

```
<?xmlversion="1.0"encoding="UTF-8"standalone="no"?>
<wsdl:definitionsxmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"xmlns:tns="http://www.exa
mple.org/serviceTrabajador/"xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"xmlns:xsd="http://w
ww.w3.org/2001/XMLSchema"name="serviceTrabajador"targetNamespace="http://www.example.o
rg/serviceTabajador/">
<wsdl:types>
<xsd:schema targetNamespace="http://www.example.org/serviceTrabajador/">
<xsd:element name="obtenerProfesor">
<xsd:complexType>
<xsd:sequence>
<xsd:elementname="id"type="xsd:int"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:elementname="obtenerProfesorResponse">
<xsd:complexType>
<xsd:sequence>
        <xsd:elementname="nombre"type="xsd:string"/>
        <xsd:elementname="apellido1"type="xsd:string"></xsd:element>
        <xsd:elementname="apellido2"type="xsd:string"></xsd:element>
        <xsd:elementname="ci"type="xsd:float"></xsd:element>
        <xsd:elementname="dpto_docente"type="xsd:string"></xsd:element>
        <xsd:elementname="grado_cient"type="xsd:string"></xsd:element>
        <xsd:elementname="categoria_docente"type="xsd:string"></xsd:element>
        <xsd:elementname="especialidad"type="xsd:string"></xsd:element>
        <xsd:elementname="ptp"type="xsd:boolean"></xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
            .
            .
            .
<xsd:complexTypename="trabajadorType"></xsd:complexType>
<xsd:elementname="insertarTrabajador">
        <xsd:complexType>
                <xsd:sequence>
                        <xsd:elementname="nombre"type="xsd:string"></xsd:element>
                        <xsd:elementname="apellido1"
                                type="xsd:string">
                        </xsd:element>
                        <xsd:elementname="apellido2"
                                type="xsd:string">
                        </xsd:element>
                        <xsd:elementname="ci"
                                type="xsd:float">
                        </xsd:element>
                </xsd:sequence>
        </xsd:complexType>
</xsd:element>
```

```
<xsd:elementname="insertarTrabajadorResponse">
        <xsd:complexType>
                <xsd:sequence>
        <xsd:elementname="vacio"><xsd:complexType></xsd:complexType></xsd:element>
                </xsd:sequence>
        </xsd:complexType>
</xsd:element>
            .
            .
            .
</wsdl:types>
<wsdl:messagename="obtenerProfesorRequest">
<wsdl:partelement="tns:obtenerProfesor"name="parameters"/>
</wsdl:message>
<wsdl:messagename="obtenerProfesorResponse">
<wsdl:partelement="tns:obtenerProfesorResponse"name="parameters"/>
</wsdl:message>
            .
            .
            .
<wsdl:messagename="insertarTrabajadorRequest">
        <wsdl:partname="parameters"element="tns:insertarTrabajador"></wsdl:part>
</wsdl:message>
<wsdl:messagename="insertarTrabajadorResponse">
        <wsdl:partname="parameters"element="tns:insertarTrabajadorResponse"></wsdl:part>
</wsdl:message>
              .
              .
              .
</wsdl:portType>
<wsdl:binding name="serviceProfesorSOAP"type="tns:serviceTrabajador">
<soap:binding style="document"transport="http://schemas.xmlsoap.org/soap/http"/>
<wsdl:operation name="obtenerProfesor">
<soap:operation soapAction="http://www.example.org/serviceProfesor/obtenerProfesor"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="serviceTrabajador">
<wsdl:port binding="tns:serviceProfesorSOAP" name="serviceTrabajadorSOAP">
<soap:address location="http://www.example.org/"/>
</wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

ANEXO 3 Herramientas pertenecientes a la suite WSO2.