



**Universidad Nacional Mayor de San Marcos**

**Universidad del Perú. Decana de América**

Facultad de Ingeniería de Sistemas e Informática  
Escuela Académico Profesional de Ingeniería de Sistemas

**Servicios web como plataforma para integración de  
sistemas de información heredados en una institución  
pública educativa**

**TESINA**

Para optar el Título Profesional de Ingeniero de Sistemas

**AUTORES**

John Blademir ANAYA VERAMENDI

Carlos HUACHO ARANDA

**ASESOR**

José Césa PIEDRA ISUSQUI

Lima, Perú

2008



Reconocimiento - No Comercial - Compartir Igual - Sin restricciones adicionales

<https://creativecommons.org/licenses/by-nc-sa/4.0/>

Usted puede distribuir, remezclar, retocar, y crear a partir del documento original de modo no comercial, siempre y cuando se dé crédito al autor del documento y se licencien las nuevas creaciones bajo las mismas condiciones. No se permite aplicar términos legales o medidas tecnológicas que restrinjan legalmente a otros a hacer cualquier cosa que permita esta licencia.

## Referencia bibliográfica

---

Anaya, J. & Huacho, C. (2008). *Servicios web como plataforma para integración de sistemas de información heredados en una institución pública educativa*. Tesina para optar el título profesional de Ingeniero de Sistemas. Escuela Académico Profesional de Ingeniería de Sistemas, Facultad de Ingeniería de Sistemas e Informática, Universidad Nacional Mayor de San Marcos, Lima, Perú.

---

Este trabajo esta dedicado a nuestra familia, en especial a nuestros hijos, que son fuente de motivación en nuestras vidas.

## AGRADECIMIENTOS

Expresamos nuestro más sincero agradecimiento al **Lic. José César Piedra Isusqui**, asesor de esta tesis, quien nos brindó su apoyo y orientación en todo el transcurso del presente estudio, igualmente por brindarnos su conocimiento y experiencia para conseguir lo mejor de esta tesis.

A los docentes de la UNMSM, por haber compartido sus conocimientos y el haber adquirido nuevas experiencias en el desarrollo de proyectos de software, incluidos en los talleres dentro de la currícula de nuestra especialidad.

A nuestros padres, familiares y a todas aquellas personas que indirectamente nos ayudaron para culminar este trabajo y que muchas veces constituyen un invaluable apoyo.

Y por encima de todo damos gracias a Dios.

# **SERVICIOS WEB COMO PLATAFORMA PARA INTEGRACIÓN DE SISTEMAS DE INFORMACIÓN HEREDADOS EN UNA INSTITUCIÓN PÚBLICA EDUCATIVA**

## **RESUMEN**

La amplia presencia de viejos sistemas informáticos y aplicaciones en las organizaciones Públicas Educativas a las que este trabajo va dirigido, como por ejemplo Universidades, Institutos, Escuelas Superiores, etc, y la necesidad de extraer información de estos sistemas heredados, hace necesario el desarrollo de nuevas herramientas para trabajar con aplicaciones no orientadas en la red a través de Internet. Obviamente la modernización de estos sistemas anticuados es deseable, pero muchas veces no es muy asequible, no sólo por razones económicas sino también por razones críticas de entrega de servicios.

Las Instituciones Públicas a inicios de sus desarrollos informáticos optaron por desarrollar sistemas en Diversas plataformas bajo el entorno diferentes, es así que uno de los entornos mas utilizados fue Fox Pro para DOS, estos sistemas por el paso del tiempo han acumulado importante información histórica en base de datos nativas dbf de Fox Pro.

Los Servicios Web son actualmente la tecnología de mayor difusión y más promisoría en el desarrollo de aplicaciones, aprovechando entre otros, su acceso por Internet y las facilidades de integración entre varios sistemas. Los sistemas de una empresa con el paso del tiempo se han ido complementando con nuevos desarrollos solicitados por sus usuarios. Bajo los parámetros de los B2B, las aplicaciones monolíticas existentes como las desarrolladas en Fox Pro no se acoplan con las estructuras apropiadas para entrar en este mundo, por lo que deben ser migradas a nuevas arquitecturas, siendo los servicios web (SeW) la opción que mejor se acomoda para este modelo.

El presente trabajo, propone una solución que permita el proceso de integración de Sistemas existentes y dándoles el tratamiento necesario para convertirlos en

Sistemas Web, lo que brinda la oportunidad de aprovechar data histórica como base para nuevos sistemas.

Palabras Claves:

Legacy System, Heredados, Integración, Migración.

# **WEB SERVICES AS A PLATFORM FOR INTEGRATION OF INFORMATION SYSTEMS RUN IN A PUBLIC EDUCATION**

## **ABSTRACT**

The extensive presence of old computer systems and applications in organizations of Public education to which this work is directed, such as universities, Institutes, colleges, etc., and the need to extract information from these inherited systems, makes it necessary to develop new tools to work with not oriented applications on the network via the Internet. Obviously modernizing these antiquated systems is desirable, but often not very affordable, not only because of economic reasons but also critic reasons in the delivery service.

Public institutions at the beginning of their developments opted to develop computer systems in different platforms under different environments, one of the environments more used was Pro for DOS, these systems by the passage of time have accumulated important historical information in dbf native database of Fox Pro.

Web Services technology are currently most widespread and most promising in application development, taking advantage among others, Internet access and ease of integration between various systems. The systems of a company with the passage Time have been complemented with new developments requested by its users.

Under the parameters of B2B, the existing monolithic applications as the developed at Fox Pro are not coupled with appropriate structures to entry into this world, so they should be migrated to new architectures, and Web services (Wse) are the option that best accommodates for this model.

This paper proposes a solution that enables the integration process of existing systems and giving them the necessary treatment to convert them into Web Systems, which provides an opportunity to exploit historical data as a basis for new systems.

Key words:

Legacy System, Inherited, integration, migration.

# ÍNDICE

|                  |     |
|------------------|-----|
| Lista de Figuras | vi  |
| Lista de Tablas  | vii |

## **CAPÍTULO 1: INTRODUCCIÓN ..... 1**

|                                                              |           |
|--------------------------------------------------------------|-----------|
| <b>1.1 ANTECEDENTES .....</b>                                | <b>1</b>  |
| <b>1.2 DEFINICIÓN DEL PROBLEMA.....</b>                      | <b>3</b>  |
| <b>1.3 OBJETIVOS.....</b>                                    | <b>4</b>  |
| <b>1.4 JUSTIFICACIÓN .....</b>                               | <b>5</b>  |
| 1.4.1 Justificación de carácter práctico .....               | 5         |
| 1.4.2 Justificación de carácter teórico .....                | 6         |
| <b>1.5 PROPUESTA.....</b>                                    | <b>6</b>  |
| 1.5.1 Principios sobre los que se funda esta propuesta ..... | 8         |
| 1.5.2 Herramientas aplicadas .....                           | 11        |
| 1.5.3 Resultados y contribuciones de este trabajo.....       | 12        |
| 1.5.4 Resultados Colaterales.....                            | 13        |
| 1.5.5 Aspectos que no incluidos en este trabajo: .....       | 13        |
| <b>1.6 ORGANIZACIÓN DE LA TESINA.....</b>                    | <b>14</b> |

## **CAPÍTULO 2: MARCO TEÓRICO ..... 16**

|                                          |           |
|------------------------------------------|-----------|
| <b>2.1 SISTEMA HEREDADO .....</b>        | <b>16</b> |
| <b>2.2 INTEROPERATIVIDAD.....</b>        | <b>16</b> |
| <b>2.3 SOFTWARE ADAPTER .....</b>        | <b>16</b> |
| <b>2.4 COMPONENTES DE NEGOCIOS .....</b> | <b>17</b> |
| <b>2.5 SISTEMAS EVOLUTIVOS.....</b>      | <b>17</b> |
| <b>2.6 INGENIERÍA INVERSA.....</b>       | <b>18</b> |
| <b>2.7 REINGENIERÍA.....</b>             | <b>18</b> |

|                    |                                                                |           |
|--------------------|----------------------------------------------------------------|-----------|
| <b>2.8</b>         | <b>LOS WEB SERVICES.....</b>                                   | <b>18</b> |
| 2.8.1              | ¿Qué es un Web Service? .....                                  | 19        |
| 2.8.2              | Características del Web Services .....                         | 23        |
| 2.8.3              | Estándares asociados al Web Services.....                      | 25        |
| <br>               |                                                                |           |
| <b>CAPÍTULO 3:</b> | <b>ESTADO DEL ARTE .....</b>                                   | <b>38</b> |
| <br>               |                                                                |           |
| <b>3.1</b>         | <b>REINGENIERÍA DE SOFTWARE E INGENIERÍA INVERSA.....</b>      | <b>38</b> |
| 3.1.1              | Algunas definiciones previas .....                             | 38        |
| a)                 | Software adapter .....                                         | 38        |
| b)                 | Componentes de Negocios.....                                   | 39        |
| c)                 | Sistemas Evolutivos.....                                       | 40        |
| d)                 | Ingeniería inversa.....                                        | 40        |
| e)                 | Reingeniería.....                                              | 40        |
| 3.1.2              | Estado de Arte de la Reingeniería Inversa.....                 | 41        |
| 3.1.3              | La adaptación como un camino de evolución .....                | 54        |
| 3.1.4              | Conclusiones.....                                              | 55        |
| <br>               |                                                                |           |
| <b>3.2</b>         | <b>ADAPTADORES DE SOFTWARE .....</b>                           | <b>55</b> |
| 3.2.1              | Conclusiones.....                                              | 58        |
| <br>               |                                                                |           |
| <b>3.3</b>         | <b>ORQUESTADORES Y LENGUAJES DE ORQUESTACIÓN.....</b>          | <b>58</b> |
| 3.3.1              | Componentes del orquestador .....                              | 59        |
| 3.3.2              | Estado del arte orquestadores y lenguajes de orquestación..... | 61        |
| 3.3.3              | Conclusiones.....                                              | 66        |
| <br>               |                                                                |           |
| <b>3.4</b>         | <b>LA INFRAESTRUCTURA INFORMÁTICA.....</b>                     | <b>66</b> |
| 3.4.1              | Tecnologías para la interconexión .....                        | 69        |
| 3.4.2              | Información adicional .....                                    | 75        |
| 3.4.3              | Conclusiones.....                                              | 76        |
| <br>               |                                                                |           |
| <b>3.5</b>         | <b>WEB SERVICES.....</b>                                       | <b>77</b> |
| 3.5.1              | Conclusiones.....                                              | 77        |
| 3.5.2              | ebXML .....                                                    | 80        |
| 3.5.3              | ResettaNet.....                                                | 81        |
| <br>               |                                                                |           |
| <b>CAPÍTULO 4:</b> | <b>DESCRIPCION DE LA SOLUCION TECNOLOGICA .....</b>            | <b>82</b> |
| <br>               |                                                                |           |
| <b>4.1</b>         | <b>Arquitectura diseño general.....</b>                        | <b>83</b> |
| <br>               |                                                                |           |
| <b>4.2</b>         | <b>Motivación.....</b>                                         | <b>84</b> |

|                                                          |                                                                                                   |            |
|----------------------------------------------------------|---------------------------------------------------------------------------------------------------|------------|
| <b>4.3</b>                                               | <b>Arquitectura conceptual de Integración de Sistemas Heredados Utilizando Web Services .....</b> | <b>86</b>  |
| <b>4.4</b>                                               | <b>Uso de la Técnica Wrapping para la Integración de los Sistemas Heredados .....</b>             | <b>88</b>  |
| <b>4.5</b>                                               | <b>Ventajas de esta estrategia de integración .....</b>                                           | <b>88</b>  |
| <b>4.6</b>                                               | <b>Descripción de la Solución Segunda Estrategia.....</b>                                         | <b>89</b>  |
| <b>4.7</b>                                               | <b>Arquitectura Física de Integración de Sistemas Heredados utilizando Web Services</b>           | <b>89</b>  |
| <b>4.8</b>                                               | <b>Implementación de Integración de Sistemas Heredados utilizando Web Services....</b>            | <b>91</b>  |
| 4.8.1                                                    | Objetivo de la Implementación .....                                                               | 91         |
| 4.8.2                                                    | Sistema Heredado .....                                                                            | 94         |
| 4.8.3                                                    | Web Services (Wrapper) .....                                                                      | 97         |
| <b>4.9</b>                                               | <b>Aplicaciones de Consumo .....</b>                                                              | <b>101</b> |
| <b>4.10</b>                                              | <b>Resumen del Capítulo.....</b>                                                                  | <b>102</b> |
| <b>CAPÍTULO 5: CONCLUSIONES Y TRABAJOS FUTUROS .....</b> |                                                                                                   | <b>104</b> |
| <b>5.1</b>                                               | <b>Conclusiones .....</b>                                                                         | <b>104</b> |
| <b>5.2</b>                                               | <b>Trabajos Futuros .....</b>                                                                     | <b>108</b> |
| <b>ANEXO 1.....</b>                                      |                                                                                                   | <b>113</b> |

## Índice de figuras

|                                                                                                      |    |
|------------------------------------------------------------------------------------------------------|----|
| FIGURA 1.1. SITUACIÓN ACTUAL DE LOS SISTEMAS DE INFORMACIÓN ..                                       | 15 |
| FIGURA 2.1. ARQUITECTURA DE WEB SERVICES.....                                                        | 22 |
| FIGURA 2.2. COMPOSICIÓN DE LA TECNOLOGÍA XML.....                                                    | 26 |
| FIGURA 2.3. WSDL DE SERVICIOS DE WEB SERVICES .....                                                  | 31 |
| FIGURA 2.4. FUNCIONALIDAD DEL SOAP.....                                                              | 33 |
| FIGURA 2.5. SOAP PARA SERVICIOS DE INVOCACIÓN DE MATRICULADOS<br>EN EL WEB SERVICES .....            | 34 |
| FIGURA 2.6. ARQUITECTURA TRADICIONAL DE UN UDDI.....                                                 | 36 |
| FIGURA 3.1. FUNCIÓN DE UN ADAPTADOR DE SOFTWARE .....                                                | 39 |
| FIGURA 3.2. MODELO DE MODERNIZACIÓN DE SOFTWARE EN FORMA<br>INCREMENTAL UTILIZANDO ADAPTADORES ..... | 48 |
| FIGURA 3.3.INTER OPERACIÓN ENTRE PARTES EVOLUCIONADAS Y NO<br>EVOLUCIONADAS .....                    | 48 |
| FIGURA 3.4 MODELO DE ADAPTADOR USANDO UN SHELL.....                                                  | 49 |
| FIGURA 3.5 MODELO DE INTEGRACIÓN TIPO RED .....                                                      | 67 |
| FIGURA 3.6 MODELO DE INTEGRACIÓN TIPO HUB .....                                                      | 68 |
| FIGURA 3.7 J2EE COMPONENTES Y CONTENEDORES .....                                                     | 72 |

# **Capítulo 1: Introducción**

## **1.1 ANTECEDENTES**

La mayor parte de los grandes sistemas de información que hasta hoy funcionan en las Instituciones Públicas Educativas del país, fueron desarrollados en la década de los ochenta bajo diferentes plataformas y tecnologías, que con el paso del tiempo, constituyen un activo importante. La amplia presencia de estos sistemas informáticos y aplicaciones en estas organizaciones, como por ejemplo Universidades, Institutos, Escuelas Superiores, etc, y la necesidad de extraer información de estos sistemas heredados, hace necesario el desarrollo de nuevas herramientas para trabajar con estas aplicaciones.

La incursión de las tecnologías relacionadas con Internet, el paradigma orientado a objetos, las bases de datos relacionales, los componentes distribuidos y la nueva mentalidad empresarial que intenta ofrecer mejores servicios a sus clientes y durante más tiempo, hacen notar la carencia de los sistemas heredados en proporcionar estas ventajas, provocando que la información que permanecía en ellos y que es totalmente aprovechable, sea objeto de diversos tratamientos para su recuperación.

En este trabajo de tesis se presenta un razonamiento de las soluciones para integrar los Sistemas Heredados (legacy systems) desde una perspectiva general y desde nuestra particular visión del problema.

“Un sistema heredado es un componente técnicamente obsoleto. Aunque la funcionalidad que un sistema heredado ofrece a los procesos empresariales puede estar disponible a través de una tecnología más moderna, la posibilidad de una interrupción del servicio durante la actualización de sistemas puede impedir una migración hacia el uso de sistemas más nuevos, o incluso la puede impedir dada la dificultad percibida en la conversión del contenido heredado para ajustarse a los nuevos modelos de contenido y formatos.”

Las necesidades de evolución y adaptación a los nuevos requerimientos tecnológicos y de negocio empujan al sistema a una nueva situación a la que no es posible llegar a través del mantenimiento clásico. En una primera aproximación, la solución a estos problemas se puede presentar a través de dos caminos: un nuevo desarrollo que incorpore nuevas tecnologías y funcionalidades o por medio de la aplicación de reingeniería al Sistema Heredado (legacy system).

Actualmente, los presupuestos de las instituciones públicas del Estado no permiten desarrollos de gran envergadura en costo y tiempo. Los Sistemas Heredados (legacy systems) constituyen una fuente valiosa de conocimiento del sistema a partir de los cuales, y una vez analizados, se puede decidir el camino a tomar para actualizar el sistema: reingeniería, abandono o una solución híbrida.

Obviamente la modernización de estos sistemas anticuados es deseable, pero muchas veces no es muy asequible, no sólo por razones económicas sino también por razones críticas de entrega de servicios.

El enfoque del problema de las aplicaciones heredadas es el de evitar cualquier modificación en los sistemas heredados que pueda poner en peligro la entrega de servicios; este enfoque también elimina la formación de los usuarios del sistema heredado al nuevo sistema, con el beneficio evidente del ahorro de costes y tiempos en la adquisición de nuevo equipamiento y el período de adaptación requerido para utilizarlo.

Un punto que debemos remarcar es que no estamos intentando transformar la aplicación entera heredada empleada internamente en una organización determinada, sino hacer que algunas de sus partes sean accesibles a los usuarios a través de la web.

La solución propuesta debería ser tan transparente como sea posible para los usuarios del sistema heredado y fácil de integrar con otros sistemas modernos.

## 1.2 DEFINICIÓN DEL PROBLEMA

En una organización, es necesaria la interoperatividad entre los sistemas heredados y sistemas implementados con tecnología moderna, debido al grado de importancia de contar con información histórica generada y almacenada en los Sistemas Heredados.

El reemplazo de los Sistemas Heredados de una forma rápida y económica, se ve impedido por restricciones como:

- Grandes inversiones acumuladas en el Sistema Legado en largo tiempo.
- Disminución del personal técnicamente habilitado.
- Pérdida significativa de conocimiento sobre las tareas internas que trabajan estos sistemas

En consecuencia, es necesario encontrar una forma de realizar este proceso cumpliendo de la mejor manera las restricciones de tiempo, costos y conocimiento adquirido.



Figura 1.1. Situación actual de los sistemas de información

### 1.3 OBJETIVOS

Proponer el uso de Servicios Web como plataforma para la integración de sistemas heredados en el contexto de una Institución Pública Educativa y plantear una solución tecnológica que sirva como estrategia al problema de interoperatividad de sistemas diseñados en diferentes plataformas y tecnologías con el fin de Automatizar los procesos dentro de la Organización.

Para lo cuál primero se analizará los sistemas de información dentro de la institución pública educativa, determinando el estado en el que se encuentran los sistemas heredados o legacy systems y los sistemas desarrollados con nuevas tecnologías. Esto con el fin de abordar la problemática vinculada a la integración de sistemas y proponer una estrategia basada en Web Services, el cual permita la interoperatividad de dichas aplicaciones, eliminando de esta manera islas de información dentro de la Organización.

Dada la necesidad de integración existente, en primera instancia a partir de requerimientos para la automatización de procesos de negocios que pasan por distintos departamentos, cada uno de ellos con su propio sistema informático. Existen por lo tanto programas o elementos de software que brindan las funcionalidades que requiere el proceso de negocios. Esos programas no están diseñados para interoperar, pero contienen la lógica del negocio que es requerida por algunas de las etapas del proceso a automatizar.

Analizar y estudiar las tecnologías e instrumentos existentes que permitan dotar a los sistemas legados de la capacidad de exportar las funcionalidades bajo las siguientes condiciones:

- Preservar la lógica de los programas que expresa las reglas del negocio.
- Mantener una única versión de código con independencia de quién y cómo lo invoque.
- No alterar su nivel de confiabilidad.
- No perder funcionalidades.

- Proponer modelos de transformación factibles, según los tipos de sistemas legados y el estado del arte.

## **1.4 JUSTIFICACIÓN**

### **1.4.1 Justificación de carácter práctico**

La necesidad de integración surge en primera instancia a partir de requerimientos para la automatización de procesos de negocios que pasan por distintos departamentos de las instituciones públicas, cada uno de ellos con sistemas informáticos modernos y heredados.

Las necesidades de evolución y adaptación a los nuevos requerimientos tecnológicos y de negocio empujan al sistema a una nueva situación a la que no es posible llegar a través del mantenimiento clásico. En una primera aproximación, la solución a estos problemas se puede presentar a través de dos caminos: un nuevo desarrollo que incorpore nuevas tecnologías y funcionalidades o por medio de la aplicación de reingeniería al sistema heredado (legacy system).

Actualmente, los presupuestos de las instituciones públicas educativas del Estado no permiten desarrollos de gran envergadura en costo y tiempo. Los Sistemas Heredados (legacy systems) constituyen una fuente valiosa de conocimiento del sistema a partir de los cuales, y una vez analizados, se puede decidir el camino a tomar para actualizar el sistema: reingeniería, abandono o una solución híbrida.

Obviamente la modernización de estos sistemas anticuados es deseable, pero muchas veces no es muy asequible, no sólo por razones económicas sino también por razones críticas de entrega de servicios.

Desde el punto de vista práctico, este trabajo sugerirá la aplicación de una Solución Tecnológica para realizar una integración de los sistemas de una institución Pública educativa. Se espera que la aportación final sirva de referencia para la integración de aplicaciones en otras instituciones que sean similares.

Con la Solución propuesta se pretende brindar la oportunidad de aprovechar data histórica y lógica de negocio de los sistemas heredados como base para nuevos sistemas eliminando también de esta manera islas de información dentro de la organización.

#### **1.4.2 Justificación de carácter teórico**

Desde el punto de vista teórico, el presente trabajo de investigación va a permitir enriquecer la concepción teórica sobre integración de Sistemas heredados, enfocados en la interoperatividad, acceso y tratamiento de información estudiando y analizando metodologías que permitan este proceso de integración y darles el tratamiento necesario para convertirlos en Servicios Web, lo que brinda la oportunidad de aprovechar lo desarrollado en el sistema legado a nivel de la lógica del negocio y al nivel de programación como base para el nuevo sistema.

### **1.5 PROPUESTA**

La Integración de sistemas legados, bajo el enfoque de esta tesina, supone la participación de elementos de programas que pertenecen a sistemas legados y que, actuando en forma integrada de acuerdo a la lógica que expresa un modelo de negocios, dan soporte al proceso que dicho modelo representa.

Existen por lo tanto programas o elementos de software que brindan las funcionalidades que requiere el proceso de negocios. Esos programas no

están diseñados para interoperar, pero contienen la lógica del negocio que es requerida por algunas de las etapas del proceso a automatizar.

El instrumento natural para especificar un proceso de negocios es el diagrama de actividades de UML. El paso siguiente es modificar el diagrama para incorporar el sistema integrador. Este diagrama contiene la información necesaria para detectar las funciones requeridas de cada sistema legado y construir un macro programa que exprese la lógica de ejecución modelada en el diagrama. El macro programa documenta el proceso de negocios y se registra en un repositorio.

Como resultado del estudio de los problemas inherentes a la integración de Sistemas Heredados en ambientes heterogéneos que se presenta una lista de los problemas acompañada de una descripción de la naturaleza de cada uno de ellos. Elementos que han sido tomados en cuenta para la propuesta de solución.

Para esta tesis se ha elegido trabajar en integración de sistemas legados en el ámbito reducido de una organización en Particular (Institución pública Educativa), en la certeza de que la automatización de procesos de negocios al interior de una organización aporta beneficios inmediatos a los usuarios. Además permite exponer los procesos de negocios, encapsulados en macro operaciones, como Web Services y por tanto contribuir a otros procesos integradores de carácter más general.

El conocimiento de la realidad informática de las Instituciones tanto estatales como privadas del país, la experiencia recogida en integrar sistemas y la poca bibliografía existente en la materia, han impulsado el desarrollo de esta tesina con una visión de conjunto de la problemática. Se entiende que este enfoque constituye un aporte interesante a la comunidad informática ya que presenta elementos aplicables tanto para la integración de sistemas, como para construir componentes a partir de sistemas legados. Ambos aspectos contribuyen a facilitar el desarrollo de soluciones para las necesidades emergentes en las áreas de “e-goverment”, “e-bussiness” y “e-commerce”. El enfoque de una metodología de desarrollo, tanto el hecho de recopilar dificultades a

considerar en todo emprendimiento cooperativo en el área que nos ocupa, como por la metodología misma constituye una guía importante para construir este tipo de soluciones.

En ese sentido se ha optado por analizar los problemas existentes, proponer una metodología de trabajo y un modelo de solución, que muestre la vinculación entre los distintos componentes necesarios para resolver los problemas, a la vez que remarcar la necesidad de reuso.

Es evidente que tanto el modelo presentado, como alguno de los componentes fundamentales (orquestador repositorios e intérprete o compilador para la gramática propuesta), requieren de una especificación detallada y de una implementación.

Los servicios ofrecidos por los sistemas legados e instrumentados mediante los procedimientos descritos como operaciones atómicas, también pueden exponerse como Web Services. Estas razones son las que han llevado a asumir una estrategia de descripción de operaciones atómicas basada en “Web Services”.

### **1.5.1 Principios sobre los que se funda esta propuesta**

- a) El encapsulamiento de funcionalidades de los sistemas legados en servicios; según M.Brodie y M. Stonebraker <sup>1</sup>, todo sistema se compone de interfaces, aplicaciones y bases de datos. En consecuencia, la interoperabilidad se puede fundar a partir de cualquiera de estos tres elementos.

Esta tesina explora la reutilización de los servicios que naturalmente ofrece todo sistema mediante interfaces, tanto de usuario como de programa. Estas, adaptadas convenientemente mediante técnicas de “wrapper” <sup>2</sup>, constituyen el mecanismo

---

<sup>1</sup> “Migrating Legacy Systems : Gateways, Interfaces and the Incremental Approach”,

<sup>2</sup> UDDI Version 3.0, [http://uddi.org/pubs/uddi\\_v3.htm](http://uddi.org/pubs/uddi_v3.htm)

para la interoperabilidad capaz de encapsular la lógica del negocio embebida en una parte del software del sistema legado.

De esta manera, cualquier sistema que necesite utilizar los servicios brindados por otro no necesita conocer sus estructuras ni sus reglas de negocios, simplemente tiene que poder acceder a los servicios que ofrece y a la descripción de los mismos en el repositorio de servicios.

b) Preservación del código que implementa las reglas del negocio; este principio apunta a varios objetivos en forma simultánea.

- La unicidad del código (no es conveniente que existan dos versiones de código, una para el sistema legado y otra para el Sistema Web Integrador).
- Asegurar idéntico comportamiento frente a identidad de datos, con independencia de su utilización en el sistema legado o en el Web Service.
- Minimizar el riesgo de introducir errores y requerimientos de testing funcional.
- Disminuir el impacto de los problemas de documentación y “program understanding”.

c) Independencia; al margen de la forma en que se instrumente la cooperación entre los distintos sistemas, es necesario que se conserve la independencia y la identidad de cada uno de los sistemas participantes.

d) Distribución de responsabilidades según competencia; el tratamiento de la información perteneciente al dominio de cada sistema, es responsabilidad del propio sistema y se hace accesible a los demás mediante los servicios que ofrece. Es interesante mencionar aquí que este criterio de encapsulamiento

condice con el concepto de “data steward” <sup>1</sup>, que viene del ámbito de los estudios sobre calidad de datos y que hace referencia a la entidad responsable por la calidad de cada dato en particular.

Si hacemos extensiva esta definición a los servicios, obtenemos una nueva propiedad: Cada sistema que exporta un servicio, se hace responsable por la definición del contenido de información vinculada a él y por que ésta se corresponda con el servicio implementado.

La aplicación de este concepto resuelve el problema que presenta el cumplir las restricciones de integridad de un sistema cuando la información de éste es manipulada desde otros sistemas, máxime si los controles de integridad están implementados en los programas y no en las bases de datos (caso bastante corriente en los sistemas legados) así como para los casos de aplicaciones que no utilizan bases de datos.

- e) La utilización de los servicios extraídos de los sistemas legados debe tener un rango de aplicación que trascienda el Sistema de Integración; la forma en que se expresen las interfaces de los servicios y macro operaciones en el Sistema de Integración Web debe contemplar la posibilidad de que los mismos puedan ser utilizados desde otros sistemas. No obstante debe reservarse para el Sistema de Integración la función de administración.
- f) El modelo debe ser independiente de productos y proveedores
- g) Se deben utilizar tecnologías independientes de los fabricantes y homologadas como estándares

---

<sup>1</sup> Mecella M, Batini C. “Cooperation of Heterogeneous Legacy Information Systems: A Methodological Framework”.

### 1.5.2 Herramientas aplicadas

A continuación se enumerarán las principales herramientas utilizadas en este trabajo:

- Técnicas de ingeniería inversa y reingeniería de software, para estudiar la manera más apropiada de transformar los sistemas legados a fin de que puedan interoperar y exportar servicios como componentes de software y respetando los principios enumerados.
- XML, como instrumento natural para el intercambio de datos.
- XMLS Schemas, para la descripción de los repositorios de macro operaciones y servicios.
- Web Services Description Language (WSDL), para describir los servicios que brindan los sistemas legados. Utilizar su lenguaje para describir los servicios, aumenta las potencialidades de reuso y no introduce limitaciones a la propuesta.
- UML, para modelar el proceso de negocios.

Muchas de las piezas de software requeridas por el modelo que se propone como parte de este trabajo, tienen una solución ya definida y adoptada como estándar, tal es el caso por ejemplo de los aspectos vinculados a seguridad en el transporte de los datos.

La infraestructura básica para otros elementos tales como autenticación de usuarios y derechos de uso, está descrita y existen productos para soportarlas. En estos casos hace falta definir los mecanismos de sincronización con los sistemas legados, lo cual no presenta demasiadas dificultades ya que una vez construido el Web Services Integrador, la administración de los usuarios se puede implementar como un servicio del CIS que es invocado por el software que administra los usuarios en cada sistema legado.

### **1.5.3 Resultados y contribuciones de este trabajo**

No se han encontrado trabajos que estudien el problema de la integración de sistemas en Instituciones públicas en el Perú en forma completa. Hay abundante bibliografía sobre métodos para adaptar software pero los ejemplos aplicados lo hacen exclusivamente para “mainframes”.

Tampoco se ha encontrado bibliografía que trate sobre metodologías para la construcción de sistemas de integración web y sus dificultades. Tampoco la hay que refiera a la gestión de un Web Service Integrador.

Hay muy poca literatura que refiera a orquestadores. Recientemente ha cobrado interés este tema, a partir de la necesidad de integración de “Web Services” y han comenzado a aparecer algunos estudios. Estos, sin embargo tratan de resolver el problema conocido como “Suplí Chain” o cadena de suministros, que presenta otras complejidades. Han surgido en los últimos meses, una serie de propuestas de gramáticas de orquestación y de coreografías con el mismo objetivo y, a nuestro juicio, demasiado dependientes de la tecnología de Web Services.

Este trabajo aporta en una serie de elementos no demasiado explorados hasta el momento:

- Como resultado del estudio de los problemas inherentes a la integración de sistemas heredados se presenta una lista de los problemas acompañada de una descripción de la naturaleza de cada uno de ellos. Elementos esto que han sido tomados en cuenta para la propuesta de solución.
- Los trabajos existentes vinculados clasificación de sistemas en base a sus interfaces citan como referencia a (M. Brodie y M.

Stonebreaker)<sup>1</sup>, que tratan solamente el caso de los “mainframes”. La clasificación realizada en esta tesis incluye la clasificación de Brodie y Stonebreaker y la amplía a otras plataformas. Esta clasificación, junto con los modelos de adaptación vinculados, puede ser utilizada tanto para integrar sistemas, como para modernizar sus interfaces de usuario o para exponerlos como Web Services.

- El desarrollo de una estrategia para desarrollo de un Sistema Integrador Web para sistemas heredados, que aporta elementos y problemas a considerar durante su ciclo de vida.

#### **1.5.4 Resultados Colaterales**

Encapsulamiento mediante operaciones atómicas y macro operaciones (dos niveles diferentes) con la consecuente capacidad de reuso a ambos niveles.

Nada impide que el enfoque desarrollado se aplique asimismo a la cooperación entre distintos programas de un mismo sistema, para obtener el proceso automatizado de un procedimiento no programado en forma convencional. La aplicación de este patrón de reuso a partir del concepto de operaciones atómicas y macro operaciones ofrece muchas posibilidades.

#### **1.5.5 Aspectos que no incluidos en este trabajo:**

Quedan fuera del alcance de esta tesis, los estudios vinculados a:

- i) La gestión de transacciones.
- ii) El diseño de los algoritmos para comprobar las propiedades de clausura de datos y flujo de ejecución.

---

<sup>1</sup> Mecella M, Batini C. “Cooperation of Heterogeneous Legacy Information Systems: A Methodological Framework”.

- iii) Los detalles de diseño e implementación de los modelos propuestos.
- iv) Los estudios que posibiliten la administración de versiones en forma automática dentro del conjunto que conforman los sistemas legados y Web Services Integradores.

## **1.6 ORGANIZACIÓN DE LA TESINA**

Todo este trabajo de tesis está estructurado en seis (06) capítulos en los cuales tocaremos temas como introducción, el marco teórico, estado de arte, propuesta de solución, etc. Como detallamos a continuación capítulo a capítulo.

El capítulo 2, se toca el marco teórico conceptual a través del cual presentamos los diferentes conceptos que serán utilizados a lo largo de nuestra investigación, es importante conocer detalladamente estos conceptos para así tener una clara comprensión del problema. Presentamos el marco teórico tanto en el aspecto tecnológico como en el aspecto de negocios, - el cual se apoyará en la arquitectura propuesta -. En la parte de negocio se toca la problemática de las instituciones públicas en nuestro país.

En el capítulo 3, presentamos el estado de arte, se centra en la clasificación del problema y el análisis y estudio de las alternativas tecnológicas de solución a nuestro problema.

En el capítulo 4, se presenta el cuerpo principal de la tesina , es decir la investigación propiamente dicha para lo cual se ha preparado un análisis propio de los tipos más conocidos de arquitecturas de tecnología de información, a saber, los de dos capas, de n capas basados en componentes. Presentando una comparativa entre sus indicadores principales. También se analiza como estado del arte las soluciones existentes en la organización y que atacan el problema de la interoperabilidad entre Instituciones públicas. Asimismo se desarrolla la metodología para la confección de arquitecturas, sobre la cual se basa la investigación. En este caso partimos de la aplicación de la metodología

Microsoft que involucra vistas de arquitectura tanto Conceptual, Lógica y Física, de la cual se tomara los elementos necesarios para nuestro caso de estudio. Por último se muestra el caso práctico de aplicación, el cual trata de la identificación y modelado de los procesos de negocios para el caso de la Integración de Sistemas Heredados

En el capítulo 5, se presentan las conclusiones y recomendaciones respectivamente de nuestra investigación.

## Capítulo 2: MARCO TEÓRICO

Para el desarrollo de este tema de tesis, es necesario conocer:

### 2.1 SISTEMA HEREDADO

Un **sistema heredado** (o sistema *legacy*) es un sistema informático (equipos informáticos y/o aplicaciones) que se ha quedado anticuado y que continúa siendo utilizado por el usuario (típicamente una organización o empresa) y no se quiere o no puede ser reemplazado o actualizado de forma sencilla.

### 2.2 INTEROPERATIVIDAD

La **interoperatividad** es la condición mediante la cual sistemas heterogéneos pueden intercambiar procesos o datos.

La interoperatividad es una propiedad que puede predicarse de sistemas de naturaleza muy diferente, como pueden ser los sistemas informáticos (en cuyo caso se le suele denotar también como *neutralidad tecnológica*).

Así por ejemplo en el campo de la informática se habla de la interoperatividad de la Web como una condición necesaria para que los usuarios (humanos o mecánicos) tengan un acceso completo a la información disponible. Entre las iniciativas recientes más destacadas para dotar a la Web de interoperatividad se encuentran los servicios Web y la Web semántica.

### 2.3 SOFTWARE ADAPTER

Básicamente la función de un adaptador de software o “adapter”, es permitir adaptar una interfaz a otra a fin de que el objeto que es adaptado, pueda colaborar con otro que requiere una interfaz diferente. Este

elemento Erich Gamma<sup>1</sup> lo presenta como “patter” estructural según se muestra a continuación.

## **2.4 COMPONENTES DE NEGOCIOS**

Cuando hablamos de componentes en realidad estamos hablando de componentes de negocios los que se definen de la siguiente forma:

Un componente de negocios es la realización de un concepto del negocio. Desde el punto de vista informático, un componente de negocios es una pieza de software concreta, que en forma independiente implementa un concepto del negocio, y que mediante la expresión de interfaces bien definidas y documentadas, es capaz de ser utilizada en forma individual o acoplada con otros componentes desde diferentes programas.

No hay una única definición de componente. Así, para Grady Booch, es una parte física y reemplazable de un sistema, que conforma y provee la realización de un conjunto de interfaces. Representa el empaquetamiento físico de elementos que de otra forma serían abstracciones lógicas, tales como clases, interfaces y colaboraciones. Mientras que, para Jim Ning, se trata de una pieza de software encapsulada, distribuible y ejecutable que provee y recibe servicios a través de interfaces bien definidas.

## **2.5 SISTEMAS EVOLUTIVOS**

Uno de los objetivos de la ingeniería de software actual es avanzar hacia el paradigma de sistemas evolutivos o con capacidad de evolución (Evolutionary Systems). En este modelo la distinción entre desarrollo y mantenimiento adaptativo es reemplazada por la noción de evolución continua del propio sistema.

---

<sup>1</sup> Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley Longman,

## 2.6 INGENIERÍA INVERSA

Ingeniería inversa es el proceso por el cual se recupera información de alto nivel a partir de bajos niveles de abstracción. Por ejemplo recuperar información de diseño a partir del código fuente u obtener un código fuente de alto nivel a partir de un programa fuente en lenguaje ensamblador (assembler)<sup>1</sup>.

## 2.7 REINGENIERÍA

Reingeniería, según Cristina Cifuentes del RER (Reverse Engineering and Reengineering Committee, de la IEEE) es el proceso de transformar un sistema al mismo nivel de abstracción. Por ejemplo, transformar un sistema escrito en lenguaje C a otro escrito en C++.

Otra definición interesante es la que adopta el SEI (Software Engineering Institute, Reengineering Center, Carnegie Mellon University): "Reingeniería es la transformación sistemática de un sistema existente en una nueva forma, que implemente mejoras en la calidad, capacidad, funcionalidad, productividad, operación y capacidades evolutivas, a bajo costo de dinero, tiempo y riesgo para el cliente"<sup>2</sup>.

## 2.8 LOS WEB SERVICES

Los *Web Services* son servicios de informaciones para *Internet*. En la actualidad, esta tecnología está siendo utilizada en un amplio dominio de aplicaciones (comercio electrónico, intercambio de información, integración de aplicaciones, etc.) para la comunicación y manejo de informaciones de diversos tipos de sistemas dentro de una organización.

---

<sup>1</sup> Cristina Cifuentes, "Reverse Engineering and Reengineering Newsletter" IEEE's Technical Council on Software Engineering TCSE Reverse Engineering and Reengineering (RER)

<sup>2</sup> Software Engineering Institute ,Carnegie Mellon University Reengineering Center "Perspectives on Legacy System Reengineering"

Los *Web Services*, se crearon con el fin de mejorar la interoperabilidad de los sistemas, haciendo uso de sus estándares, los cuales permiten un mejor funcionamiento dentro de un ambiente de *Internet*. Estos sistemas son desarrollados en múltiples lenguajes de programación y plataformas tecnológicas de aplicaciones. Los *Web Services*, soportan una colección de funciones que están empaquetadas como una unidad para ser publicadas en la *Web*, permitiendo un mejor rendimiento del mismo en las organizaciones.

### 2.8.1 ¿Qué es un Web Service?

Un Web Service, es una aplicación auto-contenida, auto-descrita, que puede ser publicada, localizada e invocada a través de la Web. Los Web Services, pueden ser invocados mediante el tipo de servicio dado, y estos son accedidos utilizando protocolos Web como: HTTP (Hyper Text Transfer Protocol) y XML (eXtensible Markup Language)<sup>1</sup>.

Utilizar HTTP, como protocolo para las invocaciones en la Web facilita el uso de la infraestructura ya existente, en la actualidad prácticamente ya esta siendo implementada en casi todas las organizaciones, para el intercambio de información y publicación de servicios del mismo. Asimismo, este intercambio es posible realizarlo sin tener que agregar más protocolos a los ya existentes en los Firewalls corporativos para poder permitir estos flujos de información, lo cual seria necesario si se utilizara algún otro protocolo<sup>2</sup>.

Los Web Services, incluyen un protocolo para el descubrimiento de los mismos <sup>2</sup>, la cual se define en un formato estándar XML con la información de descubrimiento, para que los usuarios

---

<sup>1</sup> Ignacio García, Macario Polo, Francisco Ruiz, Mario Piattini, "Servicios Web" Universidad de Castilla-La Mancha, España. (Enero, 2005)

<sup>2</sup> Raul Ruggia, Jorge Besil, Carla Pais, Dario Sande. "Interoperabilidad entre Servidores de Aplicaciones Heterogeneos" (Julio, 2003)

puedan recibir este tipo de formato, permitiendo descubrir los servicios que este presta mediante un URL (Universal Resource Locator). En los casos en los que los usuarios no conozcan el URL, también se incluye dentro del servicio un mecanismo UDDI de estándares para que los proveedores de servicios publiquen sus servicios y los consumidores los puedan encontrar.

A continuación, se definen algunas reseñas históricas que dieron inicio a la creación de los servicios de Web Services<sup>1</sup> :

- En octubre de 1994, se crea el *World Wide Web Consortium* (W3C), “a fin de liderar el desarrollo de la *Web* a su máximo potencial, desarrollar protocolos comunes y asegurar su interoperabilidad, velar por la estandarización mediante recomendaciones técnicas”. Integrada en la actualidad por alrededor de 500 miembros, el W3C define los principios fundamentales de diseño de la *Web*: Interoperabilidad, Evolución y Descentralización.
- En 1998 se adopta el “Extensible Markup Language” (XML). El XML es la forma universal para documentos o datos estructurados sobre la *Web*. En la dirección <http://www.w3.org/XML> en el documento “XML in 10 points”, la W3C expone brevemente las bases sobre las que se sustenta XML y las direcciones futuras. Hay dos factores principales que inciden en el desarrollo del XML: i) El “Business-to-Business” (B2B) y “e-Commerce” que requieren que las organizaciones que utilizan diferentes plataformas se comuniquen e intercambien datos. ii) Muchas instituciones necesitan ponerse a disposición de aplicaciones basadas en *Web*, información normalmente almacenada en sistemas de “back-office”, como bases de datos y documentos<sup>1</sup>.

---

<sup>1</sup> Jorge Abin de Maria. “Integración de Aplicaciones Encapsuladas para el Desarrollo de Sistemas de Información Cooperativos”. Instituto de Computación. Facultad de Ingeniería. Universidad de la Republica. (Noviembre, 2002)

- Gracias al respaldo de Microsoft e IBM. En Abril del 2002, se crea una nueva organización la “Web Services Interoperability organization” (WS-I) [www.ws-i.org](http://www.ws-i.org) que se define a si misma “como una organización abierta destinada a promover la interoperabilidad de los Web Services, entre plataformas, sistemas operativos y lenguajes de programación diferentes”. Esta organización trabaja con las industrias y organizaciones de estándares para satisfacer las necesidades de los usuarios como proveedores de ayudas, mejores prácticas y recursos para el desarrollo de soluciones basadas en Web Services.

La potencia de los *Web Services*, radica además de su gran interoperabilidad y extensibilidad, gracias al uso de XML, en que ellos pueden ser combinados para proveer operaciones más complejas. Varios programas proveen servicios simples que puedan interactuar, para permitir operaciones complejas. A fin de poder obtener una automatización completa de este tipo de interacciones, la arquitectura de los *Web Services* necesita mayor entendimiento y el desarrollo de algunas nuevas tecnologías.

Los Web Services<sup>1</sup>, son un conjunto de estándares basados en: XML, SOAP, WSDL, y UDDI, que permiten la interoperabilidad entre sistemas con el HTTP como protocolo de transporte. (Ver figura 2.1).

---

<sup>1</sup> Raul Ruggia, Jorge Besil, Carla Pais, Dario Sande. "Interoperabilidad entre Servidores de Aplicaciones Heterogéneos" (Julio, 2003)

|                                         |
|-----------------------------------------|
| <b>DESCUBRIMIENTO</b><br>UDDI           |
| <b>DESCRIPCION</b><br>WSDL, Esquema XML |
| <b>FORMATO DEL MENSAJE</b><br>SOAP      |
| <b>CODIFICACION</b><br>XML              |
| <b>TRANSPORTE</b><br>HTTP               |

*Figura 2.1. Arquitectura de Web Services*

En la figura 2.1, se muestra la arquitectura del Web Services, el cual esta constituido de la siguiente manera:

- **Descubrimiento:** La aplicación cliente que necesita acceder a las funcionalidades que expone un Web Services necesita una forma de resolver la ubicación del servicio remoto. Se logra mediante un proceso llamado normalmente descubrimiento (discovery). El descubrimiento se puede proporcionar mediante un directorio centralizado (UDDI).
- **Descripción:** Una vez que se ha resuelto el extremo de un Web Services, el cliente necesita suficiente información para interactuar adecuadamente con el mismo. La descripción de un Web Services implica meta datos estructurados sobre la interfaz que intenta utilizar la aplicación cliente así como documentación escrita sobre el Web Services, para ello describe sus servicios en un WSDL con un esquema estándar en XML.
- **Formato del mensaje:** Para el intercambio de datos, el cliente y el servidor tienen que estar de acuerdo en un mecanismo común de codificación y formato de mensaje. El uso de un mecanismo estándar de codificar los datos asegura que los

datos que codifica el cliente los interpretará correctamente el servidor, utilizando para ello un servicio de mensajería de petición y respuesta por medio de un protocolo estándar llamado SOAP.

- **Codificación:** Los datos que se transmiten entre el cliente y el servidor necesitan codificarse en un cuerpo de mensajes, para que estos puedan integrar aplicaciones, en múltiples lenguajes de programación y diversas plataformas tecnológicas, utilizando para ello un formato estándar XML.
- **Transporte:** Una vez que se ha dado el formato al mensaje y se han serializado los datos en el cuerpo del mensaje, se deben transferir entre el cliente y el servidor utilizando como protocolo de transporte HTTP.

### 2.8.2 Características del Web Services

Para poder integrar diversas aplicaciones que fueron construidas en distintas plataformas, el Web Services permite la integración de soluciones de negocio para una o varias organizaciones. Estos sistemas o aplicaciones que fueron desarrollados dentro de la tecnología de Web Services, apuntan a formar parte de la nueva generación de aplicaciones de Sistemas Heredados.

A continuación, se presentan las principales características del Web Services:

- **Interoperabilidad:** Cualquier Web Services puede interactuar con otro Web Services. Como los Web Services pueden ser implementados en cualquier lenguaje, los desarrolladores no necesitan cambiar de ambientes de desarrollo para producir o consumir Web Services.
- **Comunicación:** Los Web Services se comunican utilizando protocolos HTTP y XML, como ya fue mencionado anteriormente. Por lo tanto cualquier plataforma que soporte

estas tecnologías se pueden implementar y acceder por un Web Services.

- **Adaptar para reducir la complejidad:** Todos los componentes en un modelo de servicio Web son Web Services. Lo importante es la interfase que este servicio provee y no como se esta implementado, por lo cual la complejidad se reduce.
- **Fácil de utilizar:** El concepto de los Web Services es fácil de entender, porque permite a los desarrolladores crear Web Services en forma rápida y fácil.
- **Soporte a la Organización:** Todas las empresas de software importantes utilizan SOAP, e incluso están impulsando el desarrollo de Web Services. Por ejemplo la nueva plataforma de Microsoft .NET esta basado en Web Services, haciendo muy simple el desarrollo de los mismos desarrollados en ASP.NET.

Los Web Services presentan las siguientes ventajas:

- Las rutinas de los Web Services, se actualizan de forma transparente para el programador y para el encargado de mantener la aplicación.
- Mediante Web Services, se puede implementar al programa funciones imposibles de contemplar bajo el uso de rutinas de librerías, como por ejemplo, incorporar un buscador de páginas Web.
- La carga de CPU, que supone la ejecución de una rutina, desaparece al usar Web Services. La carga se reparte por Internet, sobre el servidor Web para el Web Services. Esto es un comienzo de "Computación Distribuida".
- Los Web Services, hacen uso de las mismas tecnologías que han sido atacadas en tantas ocasiones. La seguridad de una empresa puede verse comprometida. La ausencia de técnicas de seguridad estándar es un obstáculo para la adopción de la tecnología.

### 2.8.3 Estándares asociados al Web Services

A continuación, se presentan los principales estándares de apoyo a la tecnología de Web Services:

#### 2.8.3.1 XML

El lenguaje XML (*eXtensible Markup Language*), es un lenguaje de marcas (etiquetas) usados para poder describir datos dentro de un formato en particular. Estos datos pueden leerse desde cualquier tipo de aplicación construida en cualquier plataforma tecnológica de software. El XML, permite transferir datos en un formato independiente de la plataforma. Por ello, el XML es una tecnología muy utilizada para transferir datos entre aplicaciones por la Internet. Los documentos XML, almacenan los datos en forma de textos. Esto hace que los documentos en XML se puedan leer fácilmente desde cualquier tipo de aplicación ya sean de distintos tipos de plataformas.

Asimismo, XML esta estructurado de la siguiente manera:

- **DTD (*Document Type Definition*)**. Es en general, un archivo/s que encierra una definición formal de un tipo de documento, y a la vez, especifica la estructura lógica de cada documento.
- **XSL (*eXtensible Stylesheet Language*)**. Define o implementa el lenguaje de estilo de los documentos escritos para XML.
- **XLL (*eXtensible Linking Language*)**. Define el modo de enlace entre diferentes enlaces.

La tecnología XML (*eXtensible Markup Language*), se ha ido desarrollando durante el transcurso de los años, según el autor Michael:

- En los sesenta, la primera tecnología de información estandarizada y estructurada de cierta importancia fue SGML (Lenguaje de Marcado Generalizado Estándar) desarrollado por IBM.
- En 1986, el SGML fue aceptado como estándar ISO (Organización de Estándares Internacionales).
- En 1989, Tim Berners-Lee y Anders Berglund, crearon un lenguaje basado en etiquetas para documentos técnicos a fin de compartirlos en Internet, ampliado del sistema SGML llamado HTML.
- En 1996, la World Wide Web Consortium (W3C) propuso introducir el poder y la flexibilidad de SGML en el dominio de la Web, ofreciendo tres ventajas que faltaban en HTML: extensibilidad, estructura y validación.
- En febrero de 1998, la W3C introdujo su nueva tecnología de etiquetado llamado XML 1.0.

La tecnología XML, esta compuesto por una serie de tecnologías de desarrollo en XML. (Ver figura 2.2).

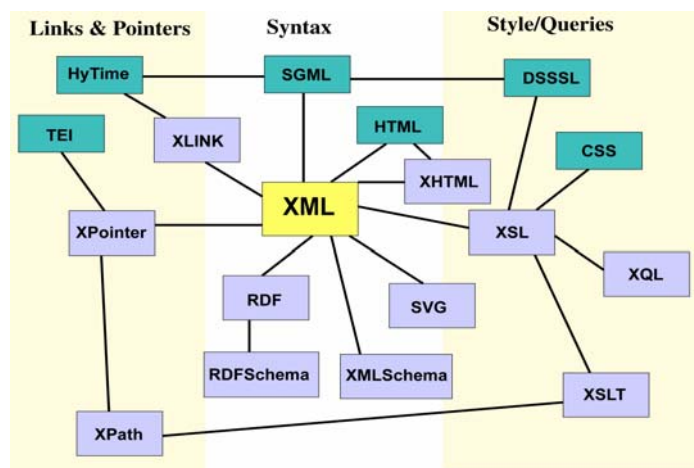


*Figura 2.2. Composición de la tecnología XML*

De esta manera, se puede definir que el lenguaje XML (eXtensible Markup Language), surge como un mecanismo para satisfacer las necesidades, para ser un formato

universal de representación de datos, que permita el intercambio de información entre sistemas heterogéneos. Con el lenguaje XML, es posible representar cualquier tipo de datos estructurados o semi-estructurados, dentro de los cuales podemos enumerar: transacciones comerciales, catálogos de libros, reportes financieros, técnicos o estadísticos, datos de tipo gráfico, anuncios publicitarios, etc<sup>1</sup>.

Un factor determinante en el suceso de la tecnología XML, es que más allá de pertenecer a un formato universal para representar datos, permite describirlos. Es decir, los datos son auto-descriptivos y pueden ofrecer todas las informaciones que los usuarios o aplicaciones necesitan para interpretar y procesar correctamente estos datos. Además, la tecnología XML pertenece a una amplia disponibilidad de herramientas de software que facilitan el rápido desenvolvimiento de las aplicaciones, haciéndose más atractiva su utilización.

El XML, es un lenguaje de marcación basado en texto, es un lenguaje que contiene elementos especiales (también llamadas de marcaciones o etiquetas) utilizados para describir la estructura y contenido de diferentes tipos de documentos, que serán utilizados principalmente por aplicaciones computacionales. Las marcaciones proporcionan significado sobre el contenido de documentos. De esta forma, documentos escritos en un lenguaje de marcación consisten de texto que representan el contenido de documentos y marcaciones que adicionan estructura e información sobre ese contenido. Ejemplos de

---

<sup>1</sup> E. Nesime Tatbul, "the Use of XML in Software Engineering" (Mayo, 2000)

lenguajes de marcación son el HTML (Hiper Text Markup Language), RTF (Rich Text Format) y PostScript.

Los lenguajes de marcación en HTML, son solo de tipo presentación, esto quiere decir que en HTML las fuentes de datos son mostradas en el Internet de manera navegable y visual, más no como una estructura de fuentes de datos de registros<sup>1</sup>.

Los lenguajes de marcación en XML son de tipo estructurado, esto quiere decir, que en la interfaz de aplicaciones y en el navegador de Internet se puede mostrar, ingresar, y también intercambiar fuentes de datos e información de sistemas.

Para ilustrar, el concepto de marcaciones y la diferencia que existe entre los lenguajes XML y HTML, a continuación se presenta un sencillo ejemplo de identificación de datos de una persona (Ver código 3.1).

- **HTML:**

---

```
<HTML>

<BODY>

<H3>Identificación </H3>

<UL>

<LI>Nombre: Lolo Fernandez</LI>

<LI>Sexo: Masculino</LI>

<LI>Fecha Nacimiento: 21/12/2000</LI>
```

---

<sup>1</sup> Libro: Michael Morrison "XML al descubierto"

```
<LI>Teléfono: 4777777</LI>
<LI>Dirección: Mayorasgo, Molina.</LI>
</UL>
</BODY>
</HTML>
```

---

- **XML:**

- < Identificación >**

---

```
<Nombre> Lolo Fernandez </Nombre>
<Sexo>Masculino</Sexo>
<FecNac>21/12/2000</FecNac>
<Telefono>4777777</Telefono >
<Dirección> Mayorasgo, Molina </Dirección>
</Identificación>
```

---

#### **Ejemplo de Libreta direcciones (HTML-XML)**

A primera vista, el lenguaje XML es similar en estilo al HTML, actualmente el XML es un súper conjunto de HTML y la mayoría de documentos HTML pueden ser considerados como documentos XML, con algunas excepciones. A XML, se le fue desarrollada como respuesta a necesidades de una generalización de lenguajes HTML. Cabe mencionar, que tanto XML y HTML son derivados del mismo lenguaje “madre” SGML (Standard Generalized Markup Language). Sin embargo, el objetivo principal de HTML es describir como los datos tienen que ser mostrados en una pantalla. Además, HTML posee un conjunto fijo de marcaciones, pero no permite

una fácil extensión de este lenguaje para atender necesidades específicas de usuarios o aplicaciones. En resumen, el lenguaje HTML, no está orientado a los datos, es solo una herramienta para presentación visual de páginas Web.

### **2.8.3.2 WSDL**

Los Web Services, utilizan el WSDL (Web Services Description Language), para especificar los tipos de contratos de cada tipo de servicio que estas realicen dentro de una sintaxis que esta basada en XML. Esta especificación permite conocer por otra parte que los consumidores utilizan métodos exportados, así como los parámetros y datos retornados.

El WSDL (Web Services Description Language), es un lenguaje de marcas (etiquetas), que definen a un Web Services. El WSDL, es un archivo de tipo estándar XML, que contiene la información sobre un Web Services. Esta información incluye los servicios Web llamados desde un Web Services, los métodos incluidos en cada uno de los Web Services y los parámetros que debemos agregarle a cada método. Además, el WSDL incluye información sobre los resultados devueltos al momento de procesar una petición de un Web Services. Por ejemplo, el WSDL define los tipos de valores devueltos por un método Web. Por ello, WSDL, es un vocabulario definido para crear un Web Services .

Además, cuando se llama a un método Web, tenemos que pasarle el nombre de usuario y la contraseña como parámetros. La información sobre el tipo y formato de los parámetros se almacena en un archivo WSDL. Cuando se procesa la petición y se devuelve el resultado, el WSDL

almacena el formato y otra información sobre el resultado devuelto.

Al momento de almacenar información sobre los métodos Web, el WSDL, almacena la información sobre el tipo de formato empleado por el usuario para poder acceder a un Web Services y especificar la ubicación en la que se encuentra disponible ese Web Services. Por ello, el WSDL describe todo el mecanismo implicado en la transferencia de datos desde un cliente de Web Services hasta el servicio de Web Services y viceversa. (Ver figura 2.3).

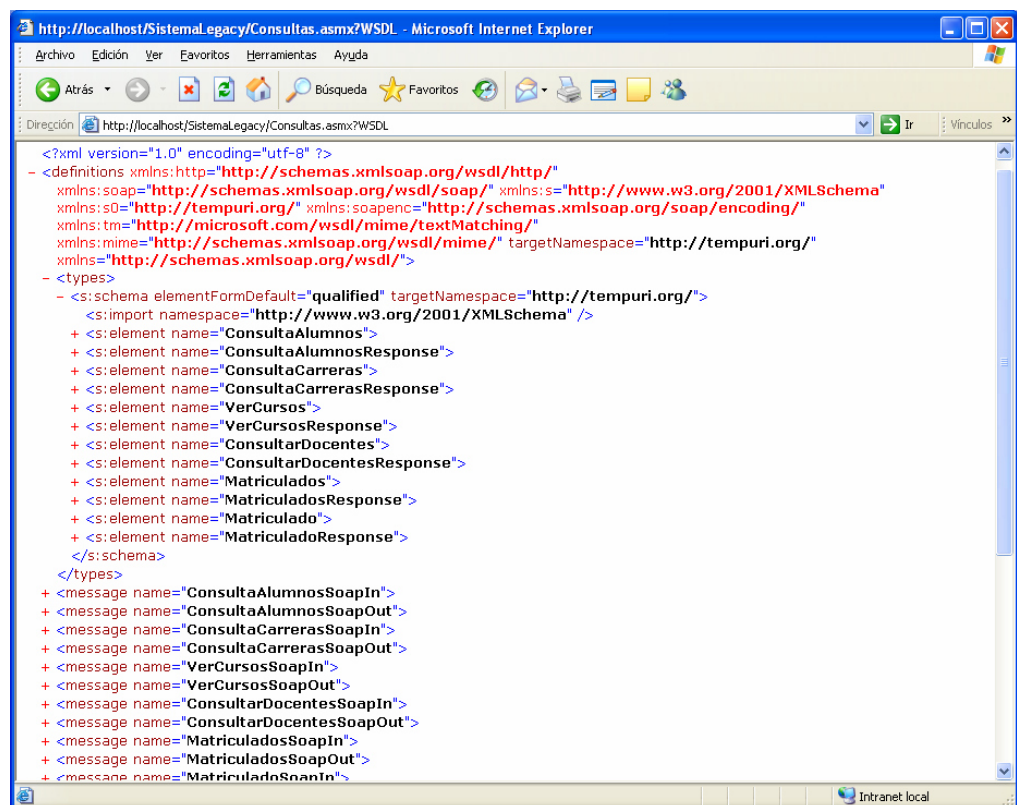


**Figura 2.3. WSDL de servicios de Web Services**

En la figura 2.3, se observa la descripción de los servicios Web de implementación de la presente tesis “Integración de Sistemas Heredados utilizando Web Services”, la cual se mostrara en capítulos posteriores. Esta información incluye los servicios Web llamados por un Web Services a

modo de prueba de servicios y presentados en un formato estándar XML.

### 2.8.3.3 SOAP

La comunicación de los Web Services, se basan en el intercambio de paquetes a través del SOAP (Simple Object Access Protocol). El SOAP, es un protocolo basado en XML, permite el intercambio de información. El SOAP, está compuesto por cuatro componentes básicos: i) un “envoltorio” que define cómo describir los mensajes y cómo procesarlos, ii) reglas para serializar (codificar) instancias de datos, iii) una especificación de cómo representar invocaciones remotas y sus respuestas, y iv) una conversión para relacionar mensajes con un protocolo de transporte (habitualmente HTTP).

Para transferir datos entre un cliente de Web Services y un servidor de Web Services y viceversa, se emplea el protocolo de transferencia SOAP. El SOAP, es un protocolo basado en estándares XML, que utiliza una aplicación de tipo cliente para acceder a un Web Services. Además del XML, el SOAP utiliza el HTTP para transferir datos. Cuando un cliente envía una petición se realiza por medio de un mensaje SOAP. El mensaje SOAP, también incluye los parámetros y las sentencias de llamada al método. Basándose en esta información del mensaje SOAP, se llama al método Web apropiado. (Ver figura 2.4).

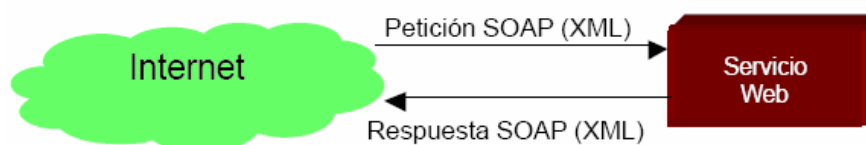


***Figura 2.4. Funcionalidad del SOAP***

En la figura 2.4, se muestra la funcionalidad del SOAP y sus procesos de petición y respuesta de mensajes de servicios del Web Services a través de la Internet.

El SOAP tiene las siguientes ventajas:

- No esta asociado con ningún lenguaje.
- No se encuentra fuertemente asociado a ningún protocolo de transporte.
- No está atado a ninguna infraestructura de objeto distribuido.
- Aprovecha los estándares existentes en la industria.
- Permite la interoperabilidad entre múltiples entornos.

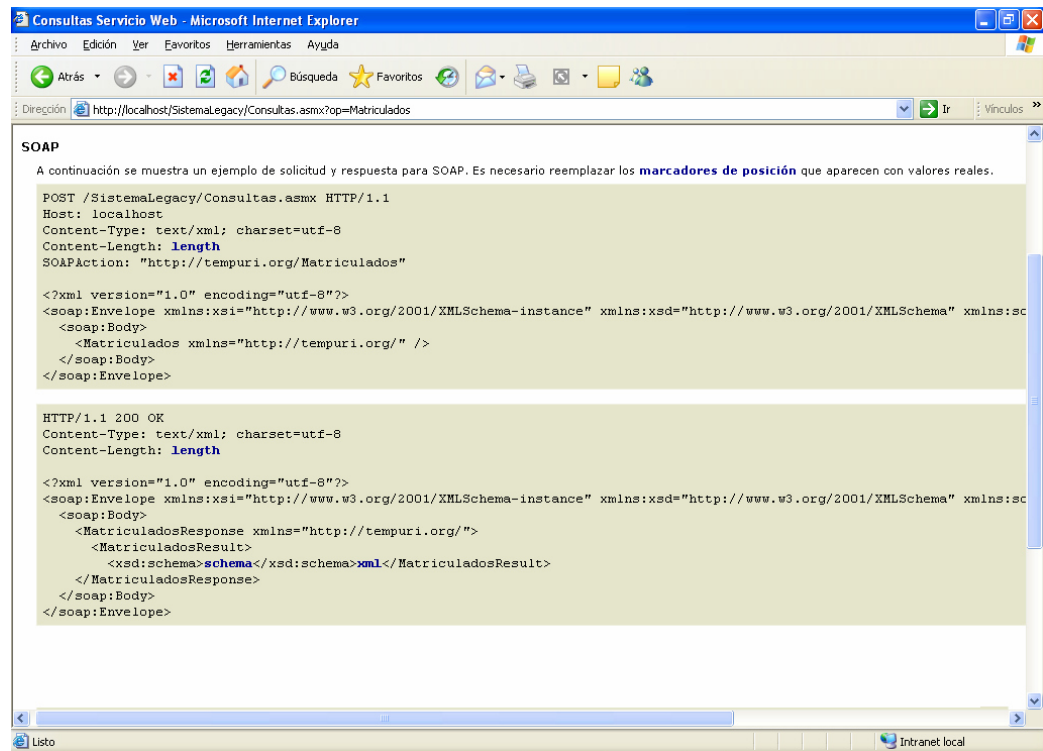


**Figura 2.5. SOAP para servicios de invocación de Matriculados en el Web Services**

En la figura 2.5, se observa la funcionalidad del SOAP, para uno de los servicios de la implementación de la presente tesis “Integración de Sistemas Heredados utilizando Web Services”, la cual se mostrara en capítulos posteriores. Esta información incluye los procesos de petición y respuesta de invocación de servicios de Matriculados llamados por un Web Services.

#### 2.8.3.4 UDDI

El Web Services, utiliza el UDDI (Universal Description, Discovery and Integration)<sup>1</sup>, para proveer un mecanismo de páginas amarillas que permitan la localización de los servicios a través de la Internet.

Cuando se desarrolla un Web Services, se tienen que registrar estos servicios dentro de en un directorio llamado UDDI. El UDDI proporciona un mecanismo para que los proveedores registren sus Web Services. Cuando se registra un Web Services, en un directorio UDDI, este crea una entrada para el Web Services. Un directorio UDDI, mantiene un archivo estándar XML por cada Web Services registrado. Este archivo XML, contiene un puntero al Web Services registrado en el directorio. Además, el directorio UDDI, también contiene punteros al documento WSDL de un Web Services. Para ello, el proveedor del Web Services tiene que describir antes el Web Services en un documento WSDL. Una vez creado este documento, se puede registrar el Web Services en el directorio UDDI .

El directorio UDDI, devuelve la lista de Web Services que se están registrando en él. El usuario podrá seleccionar entonces el método Web requerido de la lista de Web Services disponibles.

El UDDI, contiene tres tipos de páginas necesarias para la información de un Web Services. i) Las páginas blancas, contienen información sobre la organización que proporciona el Web Services. Esta información incluye el nombre, dirección y otros datos de contacto de la empresa que proporciona el Web Services, ii) Las páginas amarillas,

---

<sup>1</sup> Raul Ruggia, Jorge Besil, Carla Pais, Dario Sande. "Interoperabilidad entre Servidores de Aplicaciones Heterogeneos" (Julio, 2003)

de un directorio UDDI contienen la información sobre las empresas organizadas en forma geográfica. iii) Las páginas verdes, proporcionan la interfaz de servicio para las aplicaciones cliente que accedan al Web Services. (Ver figura 2.6).

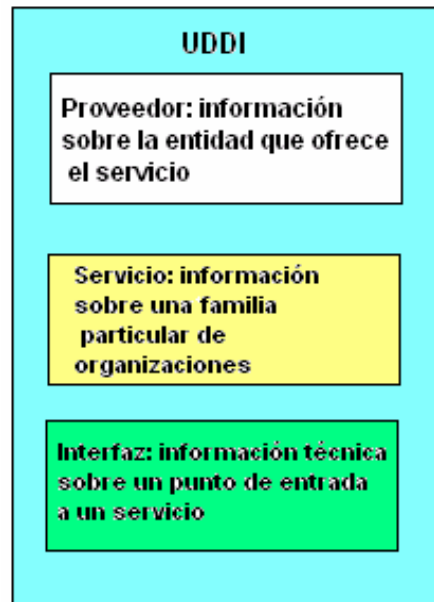


*Figura 2.6. Arquitectura tradicional de un UDDI*

El UDDI, provee de un mecanismo para que los negocios se "describan" a si mismos y los tipos de servicios que proporcionan y luego se puedan registrar y publicar en un registro UDDI. El registro del UDDI, trabaja de la manera siguiente:

#### **Definir la entrada de UDDI**

- Determina los Modelos (archivos WSDL), que utilizan las implementaciones del *Web Services*.
- Determina el nombre de la empresa y una breve descripción de la misma en varios idiomas, si es necesario, así como los contactos principales para los que el *Web Services* ofrece.

- Determina las categorías e identificaciones adecuadas para la empresa.
- Determina los *Web Services*, que la empresa ofrece a través de UDDI.
- Determina las categorías adecuadas para los servicios.

### **Registrar la entrada de UDDI**

- Una vez finalizada la tarea de definición, el siguiente paso consiste en registrar la empresa. Deberá obtener una cuenta con un registro UDDI.

### **Buscar la entrada en UDDI**

- Es recomendable realizar tres comprobaciones una vez registrada la entrada en UDDI.
- Buscar la empresa por su nombre y categorizaciones.
- Asegurarse que este referenciada a la dirección Web.
- Luego de 24h, busca el servicio en Internet.

## Capítulo 3: Estado del Arte

La integración de sistemas comprende una serie de instrumentos tecnológicos cuyo estudio es importante para conocer el estado de cada una de ellos en cuanto a su desarrollo, posibilidades que ofrecen y productos existentes en el mercado.

En particular, se realizará un estudio del estado de arte de la reingeniería de software y de la ingeniería inversa, pero se hará un estudio más detallado del estado de los Web Services en el entendido que son sumamente importante desde el punto de vista conceptual para el desarrollo de este trabajo. También se incluirá un estudio referido a adaptadores de software y orquestadores.

### 3.1 REINGENIERÍA DE SOFTWARE E INGENIERÍA INVERSA

#### 3.1.1 Algunas definiciones previas

##### a) Software adapter

Básicamente la función de un adaptador de software o “adapter”, es permitir adaptar una interfaz a otra a fin de que el objeto que es adaptado, pueda colaborar con otro que requiere una interfaz diferente. Este elemento Erich Gamma<sup>1</sup> lo presenta como “patter” estructural según se muestra a continuación.

---

<sup>1</sup> Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley Longman, 1995,

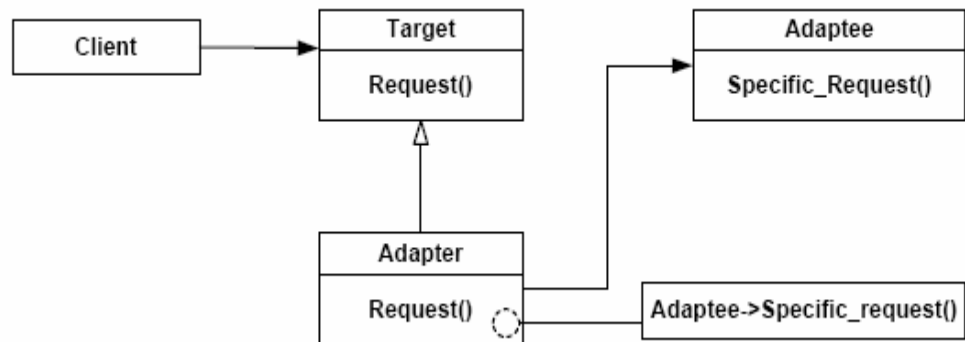


**Figura 3.1. Función de un Adaptador de Software**

|          |   |                                                                 |
|----------|---|-----------------------------------------------------------------|
| Target:  | : | Define la interfaz específica que utiliza el cliente            |
| Client:  | : | Colabora con los objetos de acuerdo a la interfaz del “target”. |
| Adaptee: | : | Define una interfaz existente que debe ser adaptada             |
| Adapter: | : | Adapta la interfaz del “adaptee” a la interfaz del “target”.    |

## b) Componentes de Negocios

Cuando hablamos de componentes en realidad estamos hablando de componentes de negocios los que se definen de la siguiente forma:

Un componente de negocios es la realización de un concepto del negocio<sup>1</sup>.

Desde el punto de vista informático, un componente de negocios es una pieza de software concreta, que en forma independiente implementa un concepto del negocio, y que mediante la expresión de interfaces bien definidas y documentadas, es capaz de ser utilizada en forma individual o acoplada con otros componentes desde diferentes programas.

No hay una única definición de componente. Así, para Grady Booch, es una parte física y reemplazable de un sistema, que

<sup>1</sup> Peter Eles, Oliver Sims, “Building Business Objects”. John Wiley & Sons, Usa 1998

conforma y provee la realización de un conjunto de interfaces. Representa el empaquetamiento físico de elementos que de otra forma serían abstracciones lógicas, tales como clases, interfaces y colaboraciones. Mientras que, para Jim Ning, se trata de una pieza de software encapsulada, distribuible y ejecutable que provee y recibe servicios a través de interfaces bien definidas.

### **c) Sistemas Evolutivos**

Uno de los objetivos de la ingeniería de software actual es avanzar hacia el paradigma de sistemas evolutivos o con capacidad de evolución (Evolutionary Systems). En este modelo la distinción entre desarrollo y mantenimiento adaptativo es reemplazada por la noción de evolución continua del propio sistema.

### **d) Ingeniería inversa**

Ingeniería inversa es el proceso por el cual se recupera información de alto nivel a partir de bajos niveles de abstracción. Por ejemplo recuperar información de diseño a partir del código fuente u obtener un código fuente de alto nivel a partir de un programa fuente en lenguaje ensamblador (assembler).

### **e) Reingeniería**

Reingeniería, según Cristina Cifuentes del RER (Reverse Engineering and Reengineering Committee, de la IEEE) es el proceso de transformar un sistema al mismo nivel de abstracción. Por ejemplo, transformar un sistema escrito en lenguaje C a otro escrito en C++.

Otra definición interesante es la que adopta el SEI (Software Engineering Institute, Reengineering Center, Carnegie Mellon University): "Reingeniería es la transformación sistemática de

un sistema existente en una nueva forma, que implemente mejoras en la calidad, capacidad, funcionalidad, productividad, operación y capacidades evolutivas, a bajo costo de dinero, tiempo y riesgo para el cliente“.

### **3.1.2 Estado de Arte de la Reingeniería Inversa**

Durante mucho tiempo las investigaciones y los desarrollos vinculados a la ingeniería inversa y a la reingeniería se realizaron al margen del interés de la comunidad informática. En la medida en que el software ha crecido en complejidad y tamaño, sumado a la existencia de sistemas cuyo código ha sido escrito a lo largo de un período de 10 a 20 años y a las carencias importantes de documentación y conocimiento de los sistemas por parte de los responsables actuales, las personas que trabajan en migración de sistemas han comenzado a prestar atención a estas prácticas.

Pero, tal como se puede apreciar en la última definición, las actividades de reingeniería no obedecen solamente a necesidades de migración; hay un aspecto de mejora y evolución que es sumamente importante.

La extracción de componentes de un sistema legado, independientemente de las motivaciones que nos muevan a hacerlo, implica de por sí, la obtención de elementos de software con un mayor nivel de abstracción.

Migrar un sistema, implica entenderlo, recuperar la documentación, recuperar o reconstruir los modelos de diseño y realizar la transformación de los programas.

La reingeniería ofrece procedimientos para migrar un sistema legado a un sistema evolutivo de una manera disciplinada. El proceso de reingeniería puede verse como la aplicación de los principios de la ingeniería a un sistema existente para que reúna los nuevos requisitos.

Desde el punto de vista de este trabajo, serán de gran utilidad los conceptos y técnicas tomadas de ambas disciplinas.

### **3.1.2.1 Consideraciones legales sobre la ingeniería inversa**

Respecto de la ingeniería inversa es importante observar que puede haber implicaciones legales respecto de cuándo es procedente o no, según la legislación de cada país. Si bien queda fuera del alcance de este trabajo, es importante mencionar que el RER (Reverse Engineering and Reengineering Committee, de la IEEE) ha auspiciado muchos trabajos que analizan este tema y en algunos países hay legislación específica. Al parecer hay una corriente que promueve amparar la ingeniería inversa cuando ésta se aplica con finalidades como las expresadas por ejemplo en la legislación Australiana<sup>1</sup>:

- Interoperabilidad, debido a la necesidad de habilitar la capacidad que otros programas puedan ser utilizados conjuntamente con el programa original.
- Para corrección de errores. En el caso de que el propietario no sea capaz de proveer una versión libre de errores en un tiempo y precio razonable, o que haya cesado su actividad.
- Para análisis de seguridad informática. El equipo en el que se desarrolla el testing está habilitado bajo ciertas condiciones limitantes, a investigar o corregir fallas que atenten contra la seguridad.

---

<sup>1</sup> Innovation, Software, and Reverse Engineering: Technological and Legal Issues "Reverse Engineering and Reengineering Newsletter"

### **3.1.2.2 Herramientas y procedimientos de detección y extracción de componentes**

Detectar y extraer componentes está estrechamente vinculado con las técnicas y experiencias asociadas a migración de sistemas legados al modelo de programación orientada a objetos.

Este tipo de migración requiere la construcción de un modelo orientado a objetos de las aplicaciones que pertenecen al dominio del sistema. La forma más evidente de obtener esto es a partir de un nuevo análisis en el que se utilizan las técnicas pertinentes al modelo. El problema es que los sistemas legados ya existen y fueron concebidos bajo otros paradigmas. Se presentan entonces tres formas posibles de realizar esta tarea.

- Reemplazar el o los sistemas legados por nuevos sistemas concebidos bajo el nuevo paradigma.
- Convertir los sistemas legados mediante técnicas de reingeniería, en nuevos sistemas expresados bajo el modelo de programación orientada a objetos. Se obtiene de esta manera un nuevo sistema que conserva las propiedades funcionales en cuanto a requerimientos. La diferencia fundamental respecto de la sustitución completa es que, en este caso, mediante técnicas de ingeniería inversa se obtiene información del sistema objeto y, a partir de la misma, se construye en forma automática la representación del sistema original bajo un modelo orientado a objetos. Luego en forma automática, semi-automática o manual, se escribe el nuevo código.
- Utilizar el código existente de los sistemas legados adecuando sus interfaces mediante wrappers y adaptadores. Hay varios “patterns” que modelan esta técnica, aplicables según el modelo de construcción de

software a utilizar. Harry Sneed, en la mayoría de los trabajos, trata con sistemas legados en ambientes de “mainframes” IBM utilizando Cics y demuestra con trabajos de campo concretos y a lo largo de una serie de años el éxito de esta técnica y de sus propios modelos (entre los años 1996 y 2002 tiene trabajos publicados referidos a este tema).

#### **3.1.2.2.1 Métodos automáticos y semi automáticos**

La ingeniería inversa, ofrece procedimientos y herramientas de software para extraer componentes de sistemas existentes. Algunas herramientas son capaces de expresar un mismo programa o sistema en otro lenguaje e incluso partiendo de sistemas construidos bajo el paradigma de programación procedural, descubrir objetos y transformarlos para expresarlos bajo el paradigma de la programación orientada a objetos.

Existe una gran variedad de herramientas complementarias, algunas de ellas comerciales, para lograr estos objetivos. Ninguna de ellas asegura la conversión de la totalidad del código en forma automática, pero pueden hacerlo en porcentajes mayores al 90%. En general se comercializan no como herramientas sino como consultoría por única vez. En el sitio web del SEI (<http://www.sei.cmu.edu/reengineering/related.html> Related Software Engineering and Commercial Web Sites ) hay una lista importante de empresas que brindan productos y servicios. El SEI ha desarrollado metodologías y herramientas apropiadas como “Minino Assets for Product lines” (MAP) y “Mining components

for a product line through Options Analysis for Reengineering “ (OAR).

Hay varias propuestas, y metodologías para la transformación/migración de sistemas que presentan características similares. Existen incluso herramientas comerciales para convertir programas Cobol en Object Cobol, como por ejemplo Visual Age for Cobol de IBM para “mainframes”, las hay también que permiten convertir de un lenguaje a otro; en este aspecto es importante mencionar a “Software Revolution Inc.”

[www.softwarerevolutionl.com](http://www.softwarerevolutionl.com), que ofrece transformación de sistemas multileguaje en forma automática. Esta empresa, que ofrece múltiples referencias de trabajos realizados tanto para organizaciones de la actividad privada como gubernamentales en los EEUU, asegura haber convertido 560.000 líneas de Cobol, con un 99.99% de certeza habiendo detectado solamente 400 “bugs”, muchos de ellos provenientes del sistema original.

El trabajo de Etzkom, L. Davis es sumamente ilustrativo respecto de las técnicas de identificación de componentes reutilizables.

Al presente y, pese a todos los esfuerzos desarrollados, todos estos instrumentos presentan una serie de dificultades tanto para su aplicación como para la verificación del resultado obtenido. Hay autores como Sneed<sup>1</sup> que

---

<sup>1</sup> Sneed Harry M., “Wrapping Legacy COBOL Programs behind an XML-Interface” Proceedings of the Eighth Working Conference On Reverse Engineering (WCRE.01) 0-7695- 1303-4/02 © 2002 \_ IEEE

citándose a si mismo y a Terekhov escribe, "... se ha probado la ineficiencia de convertir programas procedurales a componentes orientados a objetos. Aún si esto funcionara, los programadores responsables rechazarían el resultado ya que éste destruye el mapa mental que ellos tienen de sus propios algoritmos. La reingeniería clásica, en el sentido de la transformación de código fuente a código fuentes, aún mediante un lenguaje intermediario, está muerta. Esa técnica, no tiene en cuenta el hecho de que los programas son un espejo de la mente de los programadores y no pueden reingenierizarse sin que se haga lo propio con la mente de ellos."

De esta manera fundamenta su técnica de utilización del código legado existente mediante wrappers y adaptaciones, como alternativa más económica, menos riesgosa y fácil de testear. Sin duda, no es una observación de poca importancia.

Con respecto al testing, es necesario realizarlo en forma completa tal como si se hubiera realizado un proceso de desarrollo completo desde cero. El uso de herramientas para hacer la reingeniería de software en forma casi automática no deja de ser una manera de reescribir el software, sujeta además a posibles errores introducidos por las propias herramientas de traducción y por la interpretación de los equipos de desarrollo. Es importante notar, que el nuevo sistema debe cumplir las mismas funciones que el sistema original y con idéntico grado de confiabilidad. Software Revolution

indica que en algunos casos sus procedimientos de test, han detectado más “bugs” provenientes del sistema original que los introducidos por el proceso automático de reescritura propuesto por ellos.

#### **3.1.2.2.2 Métodos semi automático utilizando Adaptadores de software**

En Santiago Comella-Dorda, Grace A. Lewis, Pat Place, Dan Plakosh y Robert C. Seacord proponen un modelo de modernización de software en forma incremental utilizando “adapters”, que sigue el esquema que se muestra en la figura 3.2.

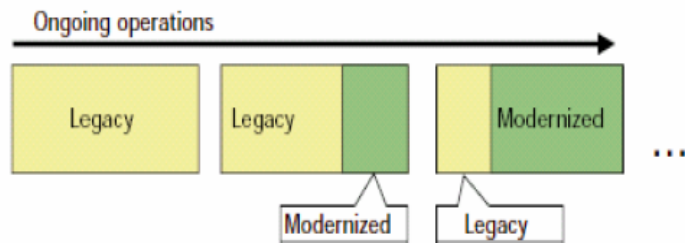


**Figura 3.2. Modelo de modernización de software en forma incremental utilizando Adaptadores**

Como se puede observar en la figura 3.2, partes del sistema modernizado coexisten con otras que aún no lo han sido; para ello se utilizan adaptadores, los cuales permiten preservar la funcionalidad del sistema durante todo el ciclo del proceso de transformación. Ya que, como se ve en la figura 3.3, mediante adaptadores, es posible lograr que una misma función sea utilizada desde dos partes del sistema en etapas diferentes de evolución. Es necesario que los adaptadores instrumenten mecanismos de adaptación bidireccionales de forma de tener una única versión del código.

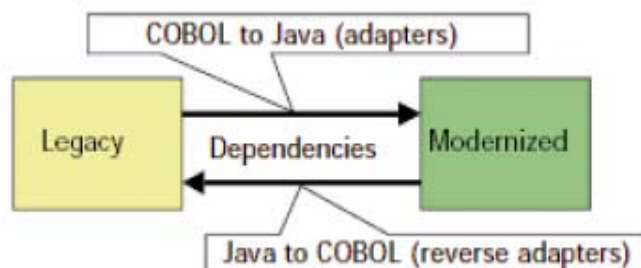


**Figura 3.3. Inter operación entre partes evolucionadas y no evolucionadas**

La funcionalidad realizada por el elemento de programa 121, ha sido re-implementada como parte del proceso de modernización. Sin embargo este elemento 121 sigue siendo invocado por el elemento 345 y a su vez invoca al elemento 129, elementos éstos que no han sido modernizados. En este modelo se construye un “shell” que interactúa con los elementos no modernizados y, mediante el “adapter”, con el componente de negocios. El “shell” asegura que las interfaces externas se mantengan y el “adapter” resuelve la interfaz entre el “shell” y los componentes, asegurando la funcionalidad y el flujo de información.

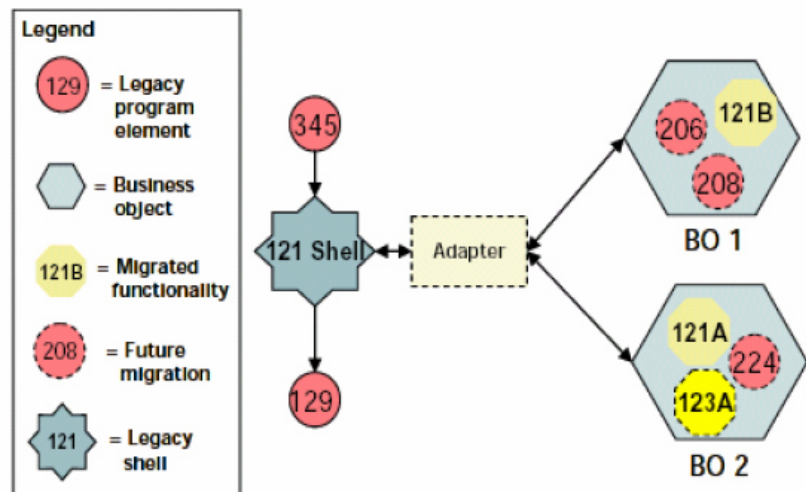


*Figura 3.4 Modelo de Adaptador usando un Shell*

Para la realización del trabajo, los autores se asistieron con herramientas de software para crear diagramas UML. El software utilizado

("SAM" System Analysis and Migration Tool), desarrollado a estos efectos, convierte elementos de programas identificados en la lista de invocaciones en clases y las llamadas a rutinas en asociaciones, nombrando cada asociación según el tipo (Perform, Call, Copy). Este modelo, a diferencia de otros que proponen la realización de proceso de transformación en forma completa, presenta facetas interesantes desde el punto de vista de este trabajo.

En primer lugar, muestra que pueden coexistir partes de software con distinto grado de evolución y funcionando hasta en plataformas diferentes. Por otra parte, estudia un modelo para detectar componentes y sus dependencias, que aporta elementos conceptuales importantes al desarrollo de la metodología que se propondrá en este trabajo.

El modelo presenta un mecanismo que habilita la coexistencia de partes de un sistema en distinto grado de evolución. Pero por otra parte reescribe completamente el código del sistema legado, lo cual generalmente presenta un grado importante de riesgo y en algunos casos no es factible por razones de documentación o entendimiento del programa.

### 3.1.2.3 Migración mediante encapsulamiento (Wrapping)

Una de las estrategias para migrar sistemas legados a sistemas distribuidos en ambientes orientados a objetos es el encapsulamiento de funcionalidades del código existente mediante el uso de “wrappers”. La idea de wrappear el código legado para ser reutilizado, es un concepto que fue introducido por primera vez por W. Dietrich en 1989 y desde entonces se ha ido ampliando y difundiendo. Actualmente, la necesidad de incorporar a internet servicios proporcionados por sistemas legados, conjuntamente con la mayor difusión de Corba, J2EE y Com, ha impulsado notoriamente esta idea. Mediante wrapping es posible adaptar el sistema legado para que sea parte de una nueva generación de sistemas, sin los riesgos inherentes a la reescritura y los altos costos de nuevos desarrollos. Es evidente que para que esto no introduzca problemas a futuro, el estado de salud del sistema debe ser “sano” o debe ser factible pasarlo a dicho estado.

Otro de los aspectos importantes de esta técnica, es que permite transformar o adaptar partes de un sistema según las necesidades y las urgencias, manteniendo el resto inalterado.

Mowbray, T. Zahari<sup>1</sup> describe seis formas diferentes de wrappers:

- Layering  
Consiste en mapear de una forma de API (application program interface) a otra de forma de mapear un conjunto de operaciones a otro completamente diferente.
- Data migration

---

<sup>1</sup> Mowbray, T. Zahari, R.: "The Essential CORBA", John Wiley & Sons, New York, 1994

Convertir o mapear un modelo de representación de datos en otro.

- Middleware

Es el caso de los wrappers para base de datos, que proveen conversión entre diversos formatos y operaciones que se expresan con dialectos diferentes. Un ejemplo es, que programas escritos para trabajar con registros operen sobre base de datos relacionales.

- Encapsulation

Separa interfaz de implementación. Puede ser utilizado para encapsular operaciones de programas a los que no se puede modificar su código.

- Wrappers for Architecture Implementation

- Wrappers for Mediators and Brokers

Estas formas de “wrappers”, según la misma referencia, utilizan siete tipos de técnicas diferentes:

- wrapping via remote procedure calls
- wrapping via files
- wrapping via sockets
- wrapping via application program interface
- wrapping via events
- wrapping via shared memory
- wrapping via macros

No existen procedimientos mágicos, parafraseando a F. Brookes, “no silver bullet”.

Muchas veces para la utilización de esta técnica se requieren algunas adaptaciones al código legado existente, lo cual significa escribir pequeños segmentos de código nuevo. En este proceso se debe poner especial cuidado en escribir el código de forma tal que no se generen dos versiones de un mismo programa. Hay varias técnicas de programación que permiten darle un comportamiento dual a

un programa para que se comporte según la interfaz desde la que es invocado.

Es también obvio que hay que realizar una serie de pruebas y verificaciones de corrección que involucran: testing individual de los diferentes componentes que intervienen y que han sido adaptados (código legado) y escritos (wrappers), test de conectividad y finalmente test de integración (Sneed <sup>1</sup>).

En virtud de que se ha detectado, en parte gracias a los trabajos de Harry Sneed, la sistematización en cuanto a la “forma” de los distintos tipos de wrappers, y considerando los casos en que se ha visto conveniente utilizar esta técnica, han aparecido una serie de trabajos en los que se muestran lenguajes de especificación de “wrappers”, con sus correspondientes compiladores; los lenguajes que hemos tenido oportunidad de estudiar, generan código para operar utilizando Corba.

#### **3.1.2.4 Utilización de técnicas mixtas**

El estado del software respecto a la documentación, conocimiento (program understanding), disponibilidad del código fuente, determina las alternativas posibles para migrar el software. La realidad no es uniforme y es posible que, dentro del mismo sistema, sobre todo en el caso de sistemas con más de una década en funcionamiento, coexistan universos diferentes. En ese contexto, es recomendable utilizar técnicas mixtas, reutilizar el código que sea posible y rescribir en forma automática o manual

---

<sup>1</sup> Sneed Harry M., Rudolf Majnar, “A Case Study in Software Wrapping”, International Conference on Software Maintenance March 1998

aquél que está en mejores condiciones de documentación y entendimiento.

### **3.1.3 La adaptación como un camino de evolución**

La adaptación consiste en hacer evolucionar solamente aquellas partes de un sistema que son necesarias para cumplir con requerimientos específicos, sin realizar cambios profundos y sin comprometer el comportamiento y la confiabilidad de los sistemas. Los casos más típicos para aplicar técnicas de adaptación responden a necesidades de cambios:

- En la presentación a los usuarios: adaptación de interfaz gráfica
- En la forma de almacenar la información: adaptación de archivos a bases de datos.
- Necesidad de intercambiar información fluida con otros sistemas: Adaptación por requerimientos de interoperabilidad.
- Encapsulamiento de funciones, para permitir su invocación desde otros ambientes y sistemas: Adaptación de funcionalidades mediante Wrapping (reuso).

Evidentemente, si en el marco de una política de evolución en la cual se marcan los objetivos tecnológicos a alcanzar se establece un diseño de la arquitectura final deseada y el estado sanitario del sistema lo permite, es posible mediante adaptaciones sucesivas evolucionar un sistema. De todas formas, es importante observar que en ninguno de los casos de adaptación que se señalaron más arriba se menciona la realización de adaptaciones al núcleo central de los sistemas ni ha sido necesario profundizar en el entendimiento del sistema, ya que todas estas adaptaciones se realizan mediante técnicas de caja negra y caja gris valiéndose del uso de adaptadores de software.

### **3.1.4 Conclusiones**

Desde el punto de vista de este trabajo, estas disciplinas han alcanzado un nivel de propuestas sumamente interesantes y en algunos casos con una importante base de experiencia acumulada, como es el caso de los adaptadores de software. Otro aspecto que ha madurado y sobre los que diferentes autores han elaborado propuestas, acompañadas muchas de ellas por experiencias de campo, es la parte metodológica, aspecto muchas veces descuidado en las tecnologías emergentes, pero fundamental para viabilizar su utilización efectiva. Finalmente, los métodos de ingeniería inversa si bien son un tanto controvertidos, no tanto por su capacidad funcional real sino por la posibilidad de adaptación por parte de los técnicos al producto obtenido, se consideran muy importantes para este trabajo. Ellos proporcionan, además del aporte conceptual, la posibilidad real de utilizarlos, no en forma masiva, ya que la propuesta no es hacer ingeniería inversa de los sistemas, sino localizada, es decir en partes concretas del software, en algunos casos pueden facilitar enormemente la construcción de adaptadores.

## **3.2 ADAPTADORES DE SOFTWARE**

En párrafos anteriores de este capítulo se analizó la técnica de encapsulamiento, conocida como “wrapping”, mientras que en el capítulo siguiente se trabajarán específicamente los “wrappers” vinculados a la extracción de componentes a partir de sistemas legados, concretamente adaptadores para programas y adaptadores para terminal (interfaz de usuario).

Respecto de los adaptadores de terminal, desde hace mucho tiempo existe la técnica de redireccionamiento de la entrada salida (Standard input, Standard output y Standard error).

Esta es, sin duda, una técnica que posibilita la construcción de un “wrapper” de interfaz de usuario a partir de la aplicación de un programa

que acepta la entrada/salida de otro programa, la interpreta y extrayendo los elementos importantes al intercambio de información, los pone a disposición de otras aplicaciones, mediante una interfaz de sistema adecuado y público.

Este esquema básico ha evolucionado conforme se ha incrementado la complejidad de las interfaces de usuario, complementándose con otras técnicas como por ejemplo la capacidad de repetir operaciones con flujo de ejecución controlado (macros) y, últimamente con lenguajes que permiten describir documentos en forma “universal” utilizando por ejemplo XML y XML Schemas, Ejb, Corba, Com, MqSeries, MsMQ, para converger en productos que pueden “dialogar” con un programa mediante su interfaz de usuario simulando un usuario e ínteroperar con otros, mediante el intercambio de documentos XML o mediante otros conectores Standard como los ya mencionados.

Si bien el esquema de redireccionamiento citado corresponde al clásico de Unix y VMS, bajo Windows es posible implementarlo utilizando una técnica llamada “Screen Scraping” o bien a partir de las “apis” de Windows.

Existen en el mercado una gran cantidad de proveedores de adaptadores de software y conectores de software (adapters y connectors), que ofrecen herramientas para realizar distintos tipos de adaptaciones.

De los más conocidos, Iway software, Attunity, Bea Software y Computer Associates ofrecen distintos tipos de adaptadores además de productos específicos para algunos de los sistemas ERP mas difundidos. Hay productos como Iway Software, que aseguran que no es necesario “tocar” el código de los sistemas legados, mientras que otros productos ofrecen convertidores de código o procedimientos para convertir el código, llegando incluso a permitir el funcionamiento dual bajo un único código. Existen otros adaptadores o conectores para los que es necesario “tocar” el código, en estos casos, las modificaciones pasan por cambiar una instrucción por una llamada a una función (ej. Read..., por Call leer...). Productores de monitores transaccionales como IBM y Bea Software ofrecen conectores como el EJB-CICS, que provee mecanismos para acceder a transacciones CICS utilizando Enterprise Java Beans, el Bea

WebLogic Java Adapter for Mainframe de BeaSoftware, IBM MqSeries Integrator Agents for Cics entre otros.

En el cuadro siguiente se presenta una lista de empresas que ofrecen software adapters. La mayoría de estos adaptadores permiten interoperar con los conectores Standard del mercado.

| <b>Fabricante</b> | <b>Producto</b>                     | <b>Tipo de adapter</b> | <b>Plataforma</b>                |                      |
|-------------------|-------------------------------------|------------------------|----------------------------------|----------------------|
| Attunity          | Application Connect                 | Aplicación             | Os400, Os390, Unix, VMS, Windows | www.attunity.com     |
| Attunity          | Application Connect                 | SDK                    | Idem                             |                      |
| Bea               | Weblogic Adapter for                | Terminar               | 3270, 5240 (os390, os400)        | www.bea.com          |
|                   | Java adapter for mainframe          | Aplicación             | CICS, IMS                        |                      |
| Multi Sofá, Inc.  | Comrad                              | Terminal - SDK         | Os400, Os390                     | www.multi-soft.com   |
| Jacada            | Integrator                          | SDK                    | Os400, Os390, Unix, VMS, Windows | www.jacada.com       |
| Iway Software     |                                     | Aplicación System      | Sap, Siebel, entre otros         | www.iwaysoftware.com |
| Iway Software     |                                     | Transactions           | CICS, IMS                        |                      |
| Iway Software     |                                     | Program – Toolkit      | C,C++,Cobol,RPG                  |                      |
| Iway Software     |                                     | Terminal               | Os390 os400, Unix                |                      |
| HostBridge        | Universal Cics Connetivity          | Transactions           | CICS-XML                         | www.hostbridge.com   |
| IBM               | MgSeries Integrator Agents for Cics | Transactions           | Os390, Windows Nt                | www.3.ibm.com        |

Existen además una serie de “frameworks” para realizar adaptaciones. Los más conocidos son los vinculados a interoperabilidad desarrollados

por Java, Corba y Microsoft, cada uno de ellos en sus respectivas plataformas. Asimismo hay un número importante de publicaciones que aportan modelos de solución para distintos tipos de problema.

### **3.2.1 Conclusiones**

Los adaptadores son una pieza fundamental para la interoperabilidad y el reuso de la lógica existente en una empresa. Es evidente que esta lección está aprendida, ya que existe un mercado importante y creciente de productos que responden a este patrón. Hay, en menor grado, un mercado de herramientas para producir adaptadores, pero, observando el surgimiento de Java Connector Architecture y de productos como Jacada, Iway, etc., este mercado seguramente se habrá de ampliar. Por otra parte, los ejemplos exitosos son abundantes lo que aporta madurez a esta tecnología.

## **3.3 ORQUESTADORES Y LENGUAJES DE ORQUESTACIÓN**

Orquestación es un término relativamente nuevo definido por analogía a la ejecución de una pieza musical por parte de una orquesta sinfónica. Así como una orquesta sinfónica no es un conjunto de músicos tocando cada uno su propio instrumento al mismo tiempo, sino que es mucho más que eso; una orquesta sinfónica en el acto de ejecutar una obra, es un conjunto de instrumentos que bajo una bien definida coreografía, la interpretan juntos de acuerdo a su arreglo musical. Cada músico debe entrar y salir a su tiempo ejecutando la parte que le corresponde en ese momento, según lo orquesta el director de acuerdo a la coreografía de la obra.

De la misma manera, un proceso de negocios debe ser orquestado bajo su propia coreografía y mediante un orquestador dirigir la ejecución de los componentes que actúan en conjunto para que, ejecutando cada cual lo

que le corresponde en el momento indicado, compongan la ejecución de ese proceso de negocios.

La coreografía proporciona secuencia y sincronización, el “qué” ejecutar y el “cuando” hacerlo. El orquestador dirige la ejecución de una cierta coreografía para articular un proceso de negocios.

### **3.3.1 Componentes del orquestador**

Para poder analizar el estado del arte de los orquestadores es necesario describir algunos de los principales componentes de la abstracción orquestador.

#### **El lenguaje de orquestación**

La expresión de una coreografía es un programa que al ser ejecutado por el orquestador invoca a los distintos servicios, según lo expresado en el programa. El lenguaje que se utilice para describir una coreografía, debe tener capacidad de expresar: flujo de ejecución condicional de servicios, ejecución de servicios en paralelo, asignación de parámetros a los servicios en tiempo de ejecución, manejo de errores propio de la coreografía y tratamiento de los errores retornados por los servicios invocados, procedimientos de interacción con el programa que solicita al orquestador la ejecución de una determinada coreografía o programa.

#### **Manejo transaccional**

El orquestador en la medida que invoca servicios que pueden alterar datos debe tener la capacidad de administrar transacciones que involucren varios sistemas, algunos de ellos transaccionales y otros no, en un entorno heterogéneo tanto de sistemas como de requerimientos de sincronismo. Cada servicio invocado es parte de una macro transacción que eventualmente tiene que ser retornada al estado previo a su ejecución (rollback). Una macro transacción

opera con transacciones que ocurren en sistemas diversos, cada uno de ellos con su propio esquema transaccional. Un servicio puede ejecutarse en forma sincrónica o asincrónica, lo cual implica que eventualmente sea necesario considerar la solicitud de “rollback” de una operación que al tiempo de solicitar esta acción no ha sido cumplido.

La característica de transacción es una propiedad del servicio heredada del sistema al que pertenece. Por lo tanto, es información a la que el orquestador debe tener acceso, pero que no necesita conocer el programador del macro programa, en consecuencia no es necesario contaminar la expresión de una coreografía con información vinculada al manejo transaccional. El orquestador debe ejecutar un “commit o rollback” en función de una condición del macro programa y de acuerdo a su conocimiento de algunas propiedades de los servicios: modificación de datos y manejo transaccional del sistema que lo ofrece.

### **Capacidad para invocar los servicios**

El orquestador según las instrucciones dadas por el macro programa debe poder invocar un servicio, el macro programa sólo debe especificarle que lo ejecute. La responsabilidad del manejo de pre-condiciones y post-condiciones está en el ámbito del macro programa el cual deberá interactuar con el orquestador para el manejo de condiciones de excepcionalidad que impliquen cancelaciones, estos aspectos se analizarán en profundidad en los capítulos siguientes. El orquestador en base al conocimiento de las propiedades del servicio deberá efectivizar la invocación, esperar o no por un resultado y manejar condiciones de time-out y transaccionalidad, entre otras.

### **Capacidad para interactuar con la aplicación que lo lanzó**

Una instancia del orquestador es activada para procesar un macro programa a partir de la acción de una aplicación la cual tiene que poder recibir una respuesta y eventualmente solicitudes de información adicional, según lo exprese la coreografía y se realice de acuerdo a los valores propios de cada instancia de ejecución. Nuevamente al programador no le interesa conocer los detalles de ejecución del macro programa, simplemente necesita los elementos propios del contrato de software y la forma de interactuar con el.

### **El registro del macro programa**

Es simplemente la capacidad para registrar las propiedades de cada macro programa, necesarias para poder ser utilizado desde una aplicación haciendo abstracción de los detalles de su implementación.

### **La descripción de los servicios y el registro**

Constituye la capacidad para registrar los servicios y sus propiedades. Deben implementar funciones de búsqueda (discovery), administración, consulta, activar/desactivar, gestión de versiones.

### **Seguridad y control de acceso**

Se requieren instrumentos para asegurar la privacidad y seguridad de la información. Para ello es necesario contar con mecanismos de control de acceso, autenticación de usuarios y sistemas, encriptación de datos.

## **3.3.2 Estado del arte orquestadores y lenguajes de orquestación**

Los productos que cuentan con instrumentos que realizan la función de orquestación los instrumentan mediante elementos propietarios y están atados al producto que los ofrece.

No hay estándares de lenguajes de orquestación ni de repositorios para sus especificaciones.

Desde el punto de vista de productos de software existentes en el mercado que actúan en forma independiente de servidores de aplicaciones, solo se encontraron dos que no focalizan su actividad en Web Services y se basan en cambio en tecnologías de interoperabilidad como J2EE, Corba y Com. Existen otros productos que necesitan de un servidor de aplicaciones específico o son parte de alguno de ellos. Este último tipo de productos son igualmente escasos y se detallan en el capítulo de infraestructura.

El concepto fundamental para hacer esta separación (requieren de un servidor de aplicaciones o no), es que los sistemas legados no necesariamente utilizan servidores de aplicaciones y no es imprescindible utilizarlos para constituir un Sistema de Información Cooperativo.

En el primer grupo se ubican los productos siguientes:

**FuegoTech:**

Producto: Fuego4 <http://www.fuegotech.com>

Reseña: Capacidad de interactuar en con múltiples ambientes (Corba, J2EE, COM/DCOM, CICS, Web Services, entre otros) y sistemas operativos.

Coreografía y orquestación: Herramienta gráfica y lenguaje de script tipo

4gl. Motor propio

Capacidad para extraer componente de sistemas legados y adaptarlos.

Manejo transaccional: No específica.

Manejo de múltiples versiones.

**Intalio:**

Producto: The Business Processes Management System (BPMS)

<http://www.intalio.com/index.esp>

Reseña: Capacidad de interactuar en con múltiples ambientes (Corba, J2EE,

COM/DCOM, CICS, Web Services, entre otros) y sistemas operativos.

Coreografía y orquestación WSCI y BPEL, y generador de código den

forma automática. Motor propio.

Manejo transaccional: JTA y OTS

Plataforma base: J2EE

Versiones: No especifica.

Extracción de componentes: No especifica

### **Las gramáticas para la expresar un proceso de negocios**

A partir de las necesidades derivadas del comercio electrónico y de la evolución de los “web services”, comienzan a presentarse propuestas de lenguajes para describir la coreografía de los procesos de negocios. La mayoría de estas propuestas contemplan complejidades derivadas del flujo de un proceso de negocios con múltiples puntos de interacción, que requieren la intervención de varias personas y tiempos de espera por acciones que puede expresarse hasta en días. El caso típico es el problema de “la cadena de suministros” (Supply Chain). Este tipo de problema es típico de los sistemas de “workflow” y no corresponden al ámbito de aplicación de este trabajo.

Una macro operación concebida dentro de un CIS (según la concepción definida para este trabajo), encapsula el proceso de negocios que atiende y presenta interfaces perfectamente definidas de modo que es factible su participación como integrante de un sistema de “workflow”.

Las gramáticas propuestas para ser homologadas por la W3C son “Xlang” utilizado por el producto Biz Talk de Microsoft, “WSFL” (Web Services Flow Language) de IBM, Sun, BEA, SAP, e Intalio introdujeron un tercer candidato el “WSCI” (Web Service Choreography Interface) y existen otras como BPML (Business Process Markup Language) de Intalio y “BPSS” (Business Process Schema Specification) de ebXML.

A principios de agosto de 2002, Microsoft, Bea e IBM presentaron el “BPEL4WS” como una gramática para la orquestación y en forma adicional proponen o sugieren “WS-transaction” y “WS-coordination”, como sub-lenguajes.

Es sumamente importante observar, las fechas y el estado actual de estas propuestas:

- *BPML working draft June 24, 2002.*
- *Web Services Choreography Interface (WSCI) 1.0, BEA, Intalio, Sun, SAP et al, June 2002.*
- *Web Services Coordination (WS-Coordination) BEA Systems, International Business Machines Corporation, Microsoft Corporation 9 August 2002*
- *Workflow Process Definition Interface -- XML Process Definition Language. Document Number WFMC-TC-1025 Document Status - XPDL 1.0 beta. (July 30, 2002)*
- *BPEL4WS (Business Process Execution Language for Web Services) Version 1.0 Aug 9, 2002.*

Un estudio comparado entre alguna de estas, realizado por Rober Shapiro fue publicado por la ebXML<sup>1</sup>.

Ambas organizaciones de estándares (Oasis y W3C), deberán ponerse de acuerdo y adoptar uno. Sin perjuicio de ello tanto Microsoft como IBM, han anunciado que la próxima versión de sus productos, Biz Talk y Websphere respectivamente, implementarán “BPEL4WS”. Por su parte el consorcio que presenta WSCI (se pronuncia whiskey), lo anuncian como producto libre, mientras que no hay una definición en cuanto a políticas de royalties para el BPEL4WS todavía.

Basados en que un protocolo compatible debe expresarse en una lengua común, se adopta el XML para expresar estas gramáticas.

---

<sup>1</sup> Robert Shapiro, A Comparison of XPDL, BPML and BPEL4WS - Cape Visions, <http://xml.coverpages.org/Shapiro-XPDL.pdf>, Published by ebPML.org. 'Rough Draft' version 1.4, August 27, 2002.

Sin embargo es imposible para una persona entender la coreografía de un proceso de negocios, más o menos complejo, escrito en XML. En consecuencia, tanto para escribir un programa como para leerlo y entenderlo, se necesita una herramienta de software adecuada. Microsoft que actualmente utiliza Xlang para describir coreografías, está ensayando C# para implementarlas. Los programadores se sienten mucho más cómodos escribiendo en este lenguaje que utilizando Xlang.

El XML presenta dificultades a la hora de entender lo que se escribe y es particularmente difícil analizar y seguir un flujo de ejecución expresado en este lenguaje. Por otra parte, la necesidad del uso de XML, es relativa, una cosa es la gramática en si misma, necesaria para expresar un proceso de negocios, y otra la forma de publicar y distribuir el programa que lo expresa.

Al tiempo en que se realizó este trabajo, la mayoría de estas gramáticas no habían sido propuestas, también es esperable que pase algún tiempo antes de que se promueva una de ellas como estándar.

En este trabajo, se explora una gramática que no se escribe en XML y que expresa solamente el flujo de ejecución, las propiedades de los servicios, tales como transaccionalidad y sincronismo entre otras, surgen de la especificación del servicio. Es el orquestador que al ejecutar el programa en conocimiento de las propiedades de cada servicio, actúa en consecuencia. Como contrapartida es extremadamente simple y puede escribirse y leerse sin necesidad de un software específico. En el capítulo V se desarrolla la misma y se presenta un ejemplo, en el Anexo III se encuentran sus especificaciones completas.

### **3.3.3 Conclusiones**

La falta de literatura, de estándares y la escasez de ofertas de productos específicos en el mercado, indican falta de madurez de esta tecnología.

La existencia de productos que no requieren de servidores de aplicaciones para orquestar operaciones de sistemas legados, muestra que uno de los conceptos manejados en este trabajo es factible.

Las gramáticas propuestas para ser homologadas, presentan una fuerte dependencia con WSDL o ebXML.

## **3.4 LA INFRAESTRUCTURA INFORMÁTICA**

No existe una definición formal para infraestructura informática o “IT infrastructure” de acuerdo a su denominación en Inglés. Es posible definirla como el conjunto de recursos y capacidades que dispone una organización, los que constituyen el fundamento sobre el cual se desarrollan y soportan las aplicaciones informáticas que sustentan los procesos de negocios.

Los instrumentos tecnológicos los podemos clasificar según su función o según la adhesión a una plataforma específica.

Mientras que los esquemas de integración básicamente pueden clasificarse en dos modelos:

El modelo de red, en el que la conexión entre las aplicaciones es una a una, Fig. 3.5, para lo cual se requiere que cada pareja de aplicaciones disponga de adaptadores para poder conectarse con su par, con independencia de los esquemas tecnológicos y adaptadores que utilicen las restantes combinaciones. Este modelo, si bien se puede aplicar rápidamente se transforma en difícil de administrar y mantener. Cada operación que involucre más de una operación tiene que ser instrumentada en una aplicación (la que la invoca) y se torna difícil el reuso. Cualquier cambio que se realice en una aplicación y que afecte a

la integración, Ej. Cambio de una interfaz, impacta directamente sobre las restantes aplicaciones involucradas.

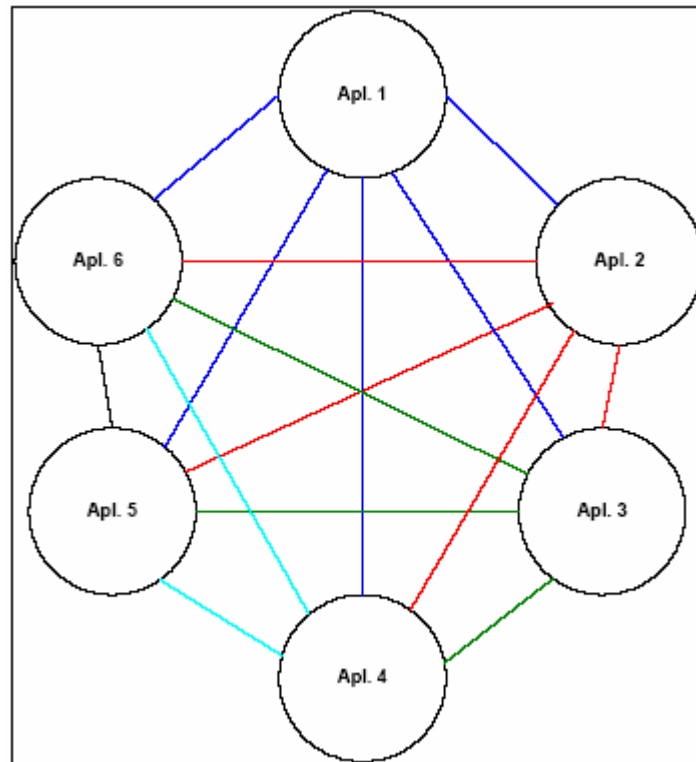


*Figura 3.5 Modelo de integración tipo Rred*

Hay otros aspectos de difícil resolución en este esquema, como por ejemplo, el manejo de transacciones globales.

Como se puede apreciar, en el diagrama de la figura 3.5, la cantidad de conexiones se pueden calcular como  $C_m^2$ , en el caso de la figura 3.5 son 15, es interesante comparar con la cantidad de conexiones que aparecen en la figura 3.6, en que se representa el modelo tipo “Hub”.

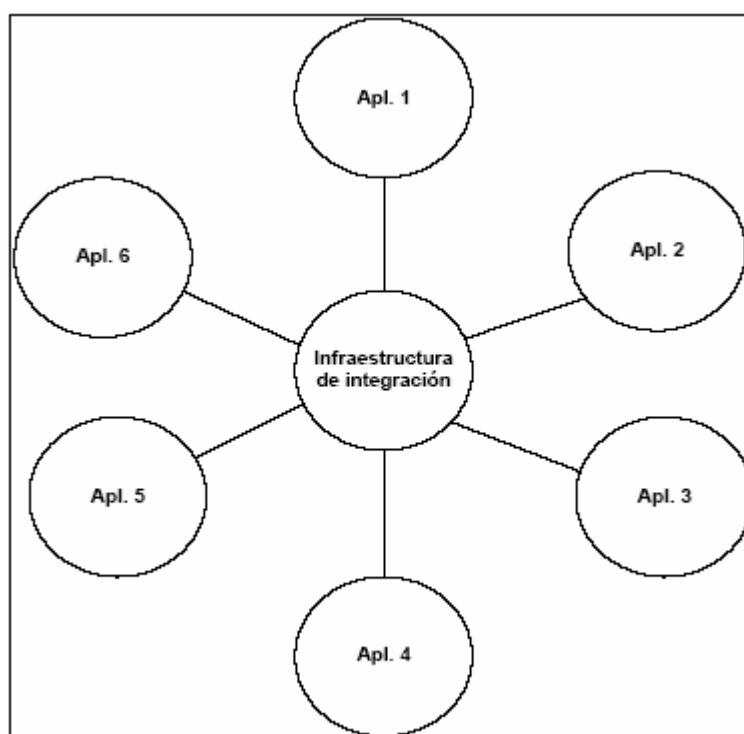


**Figura 3.6 Modelo de integración tipo Hub**

Con el surgimiento de los modelos tipo “Hub”, comienzan a proliferar los “middleware”. En este modelo, cada aplicación se conecta a un hub, en el que residen todos los elementos de integración requeridos: “brokers”, soporte de mensajes, el repositorio de las operaciones o servicios disponibles, repositorio de macro operaciones, el motor de orquestación y el soporte de conectores y manejo transaccional, como los servicios más importantes.

A los efectos de este trabajo tienen particular interés las tecnologías para la integración de aplicaciones. En forma general dichas tecnologías se pueden agrupar en: i) Enterprise Application Integration (EAI), se trata de productos y “frameworks” para facilitar la integración de aplicaciones y ii) los que englobaremos bajo el nombre de “web services”, que son los instrumentos que permiten la interoperabilidad a nivel de Web. Si bien este estudio las presenta separadas en grupos diferentes, en la práctica los productos que se aplican para EAI extienden sus funcionalidades para soportar “web services” y con ellos interoperabilidad a nivel de Internet.

El objeto de este estudio no es analizar cada una de las alternativas existentes en cada grupo, sino simplemente hacer una breve referencia a las más representativas junto con un resumen a modo de conclusiones sobre el estado del arte. Se presenta además un cuadro de productos para la integración de aplicaciones.

### **3.4.1 Tecnologías para la interconexión**

En 1989 nace el Object Management Group (OMG), organización integrada actualmente por alrededor de 800 empresas, cuyo objetivo es desarrollar propuestas tecnológicas independientes de los fabricantes y dirigidas a facilitar la interoperabilidad. *“El Object Management Group (OMG) es un consorcio sin fines de lucro, cuya membresía es abierta, que produce y mantiene especificaciones para la industria informática, dirigidas a la interoperabilidad entre las aplicaciones empresariales”*, Richard Soley (Chairman y CEO de OMG) en la presentación de la nueva propuesta de OMG, el “Model Driven Architecture” (MDA), afirmaba *“Durante once años la OMG se ha puesto el foco en asegurar que se puede integrar lo construido, con lo que se está construyendo, con lo que se va a construir...”* Si bien los esfuerzos de la OMG han sido y continúan siendo muy importantes, su propuesta no ha logrado imponerse como el único estándar para la interoperabilidad.

La puja entre los fabricantes de software, el “time to market”, ha conspirado para facilitar esta tarea. La intención de ofrecer “ventajas adicionales” o adelantarse a que se homologuen propuestas, alegando que la OMG es lenta. En suma, la disputa entre algunas empresas por prevalecer en el mercado, tiene como consecuencia que a la fecha existan tres propuestas de infraestructura para la interoperabilidad, ellas son: OMG, Java J2EE y Microsoft .NET. Si bien existe una mayor aproximación entre las propuestas de Java y la OMG, que la que existe entre OMG y Microsoft, todos apuestan a la interoperabilidad y por consiguiente

incluyen elementos que permiten interactuar con las otras plataformas, pero falta mucho para que esto sea realmente sencillo y estable.

#### **3.4.1.1 La propuesta de Microsoft**

La infraestructura de .Net y Biz Talk Server:

Biz Talk Server es un producto que Microsoft lo define como un “application integration server”, diseñado para realizar procesos de negocios complejos en ambientes distribuidos.

Se núcleo central se compone de un orquestador y un broker. El orquestador se compone de una herramienta gráfica que facilita el diseño del flujo de operaciones, el cual se expresa en XLANG, un lenguaje para la descripción de “workflow” basado en XML.

Para completar los servicios de integración, Microsoft define un “framework” de integración que se compone de Biz talk Server más un conjunto de herramientas adicionales, todos ellos del mismo proveedor. Este producto dispone de una cantidad importante de adaptadores de software ya construidos, pero no ofrece una herramienta para desarrollar nuevos.

Desde el punto de vista de transporte, sólo puede utilizar http, soap y mime; estas restricciones introducen un “overhead” innecesario si se utiliza para integrar aplicaciones en ambientes de red local o de una red privada.

Para interactuar con otros servidores, éstos deben ser BizTalk Framework Compliant (BFC) Server. Un BFC Server está representado por un conjunto de servicios que proveen la funcionalidad “message-processing” definida en las especificaciones del “framework” de BizTalk.

Una especificación completa de esta arquitectura se puede obtener en

<http://www.microsoft.com/ntserver/techresources/appserv/overview.asp?bPrint=True>

Si bien es un producto de integración y dispone de instrumentos como para actuar en ambientes heterogéneos, todos los componentes deben ser instalados sobre plataforma Microsoft. Sin embargo, es importante destacar que la iniciativa de Microsoft .Net está siendo portado a otras plataformas, esto seguramente tendrá impacto sobre el área de aplicación de Biz Talk Server.

<http://www.devx.com/free/hotlinks/2002/ednote061902/ednote061902-1.asp>

#### **3.4.1.2 La propuesta J2EE**

La plataforma J2EE propuesta y liderada por Sun Microsystems, está acompañada con un compromiso muy fuerte de por lo menos otras 15 empresas entre las que se encuentran IBM, Bea, Oracle, SAP, Borland, Sybase, sumados a los aportes del "Open Source" que aportan productos y propuestas. Sobre esta plataforma base, existen varios productos de integración de aplicaciones. Una matriz de comparación entre servidores de aplicaciones está disponible en:

<http://www.flashline.com/components/appservermatrix.jsp>.

A los efectos de asegurar la compatibilidad de los distintos productos se ha puesto a disposición de la comunidad un par de herramientas importantes: Un test de compatibilidad y un "benchmark".

"Write Once, Run Anywhere™", es el concepto fundamental de la plataforma Java, la portabilidad, interoperabilidad, acceso a datos y recursos de las

aplicaciones legadas, seguridad y componentización de las aplicaciones son las banderas de esta plataforma. La figura 3.7 muestra los principales tipos de componentes y sus contenedores. Este diagrama no ilustra la totalidad de los componentes y contenedores que existen en la actualidad. Documentación adicional se puede obtener en: <http://java.sun.com/j2ee/>, <http://java.sun.com/j2ee/white/> y <http://java.sun.com/blueprints/>. Una lista completa de productos certificados J2EE se obtiene en: <http://java.sun.com/j2ee/compatibility.html>. Información sobre servidores de dominio público está disponible en: <http://www.jboss.org/>. Una revisión sobre los productos en: <http://www.theserverside.com/reviews/index.jsp>

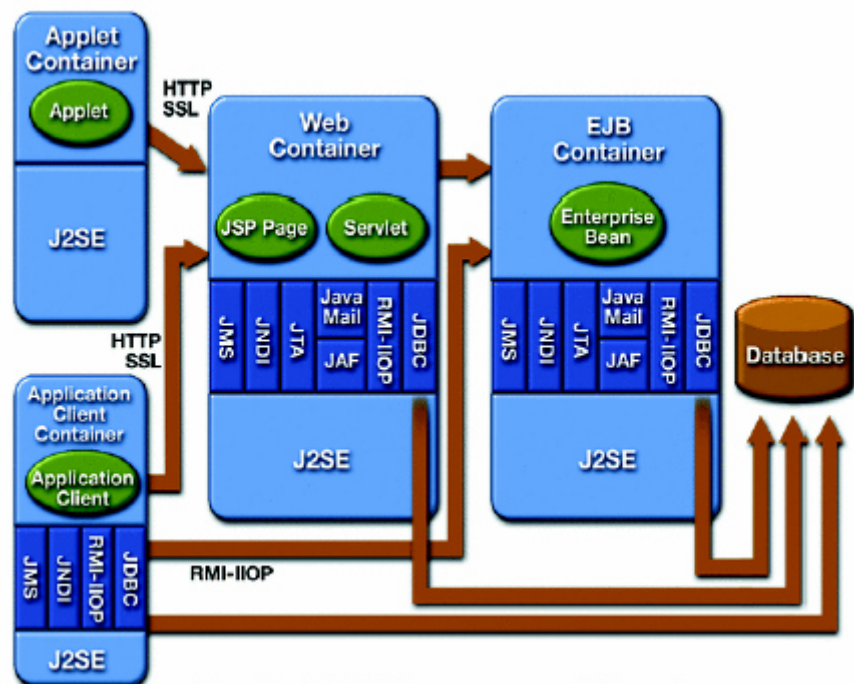


**Figura 3.7 J2EE Componentes y Contenedores**

### 3.4.1.3 OMG

En setiembre de 2001, la OMG acorde a su propuesta fundacional de aportar a la interoperabilidad y comprendiendo que desafortunadamente existen hoy en día múltiples estándares de la industria, Corba, EJB/J2EE, .NET, XML/SOAP y que muchos otros habrán de venir; la OMG propone el “Model Driven Architecture” (MDA) como un paraguas bajo el cual construir aplicaciones que permanezcan con independencia de las tecnologías existentes y futuras, sobre las que operen o puedan llegar a operar.

*“Visto que las redes y computadores se tornan cada vez más accesibles y más potentes, los estándares de interconexión deben evolucionar. Constantemente aparecen nuevas tecnologías, nuevas aplicaciones sobre nuevos nichos de negocios. El XML es un claro ejemplo de esto. ¿Cómo pueden las organizaciones asegurarse de que sus sistemas de información de misión crítica serán aplicables a los nuevos estándares y se adaptarán a las nuevas capacidades de hardware y software?”<sup>1</sup>.*

La respuesta de la OMG es el MDA. El “Model Driven Architecture” es una nueva forma de escribir especificaciones y desarrollar aplicaciones, cubre todo el ciclo de vida, diseño, desarrollo, integración y gerenciamiento de aplicaciones y datos mediante el uso de estándares abiertos. Se basa en el “Plattaform-independent model” (PIM). Una especificación completa de MDA consiste de un modelo basado en UML, el modelo independiente de la plataforma definitivo, y uno o más modelos específicos para cada plataforma: “Plattaform-spceficic model” (PSM) el cual es un conjunto de

---

<sup>1</sup> Andrew Watson, Model Driven Architecture -An Executive Overview, Object Management Group Vice President and Technical Director. - Datamation , The engeneering data newsletter May. 2002.

definiciones de interfaz de “middleware” que se desee soportar.

Basado en: UML, el único lenguaje de modelado propuesto hasta el momento, XMI (XML Metadata Interchange) es el estándar para intercambio de modelos en XML y en Corba, el único “middleware” estándar independiente de los fabricantes, la principal virtud de MDA es separar la especificación lógica de la plataforma específica de “middleware” sobre la que se vaya a implementar. La OMG propone que las empresas utilicen MDA para expresar sus objetivos de integración a nuevas plataformas como forma de preservar su inversión en los sistemas actuales.

Los principios sobre los que se basa esta propuesta son <sup>1</sup>:

- **Portabilidad**, aumentar el reuso para reducir el costo y complejidad del desarrollo y gestión de las aplicaciones, tanto ahora como en el futuro.
- **Interoperabilidad entre plataformas**, utilizando métodos rigurosos para garantizar que los estándares basados en múltiples tecnologías todos ellos implementen en forma idéntica las funciones del negocio.
- **Independencia de la plataforma**, reduciendo en gran forma los tiempos, costos y la complejidad asociada a llevar aplicaciones sobre plataformas diferentes, incluyendo aquellas que ya han sido introducidas (no solamente las futuras).
- **Especificidad del Dominio**, utilizando modelos específicos para el dominio de aplicación (ERP, CRM, Transporte aéreo, genoma, etc.), es posible incrementar

---

<sup>1</sup> Jon Siegel, Developing in OMG's New Model-Driven Architecture - Director, Technology Transfer, Object Management Group - Datamation , The engineering data newsletter May. 2002.

la velocidad para implementar aplicaciones específicas a un dominio particular sobre diversas plataformas.

- **Productividad**, facilitando a los diseñadores, desarrolladores y administradores de sistemas, la utilización de lenguajes y conceptos que les son comunes, estos se sienten más cómodos y se mejora la comunicación entre los equipos.

En el trabajo de Jon Siegel es muy ilustrativo en el uso de este nuevo instrumento.

Es importante mencionar que MDA se construye sobre los otros instrumentos estándar de la OMG: Corba, MOF (MetaObject Facility), UML- Omg's Unified Modeling Language, XMI – XML Metadata Interchange, y CWM – The Common Warehouse Metamodel. Información detallada de las especificaciones y del estado de desarrollo de de estos productos se encuentra en [www.omg.org](http://www.omg.org).

En el corto tiempo que lleva MDA de su lanzamiento, si bien no está demasiado difundido, hay importantes fabricantes de software que han adoptado o apoyado esta propuesta.

Una matriz de comparación para productos que utilizan Corba se puede encontrar en:

<http://www.puder.org/corba/matrix/>.

### 3.4.2 Información adicional

En la dirección siguiente hay información detallada sobre los distintos elementos que conforman cada una de las tecnologías mencionadas, así como también comparaciones sobre instrumentos adicionales, como servicios de mensajería.

<http://my.execpc.com/~gopalan/index.html>.

La disputa entre Microsoft y Java es muy fuerte, es posible aprender sobre el estado de cada una de las propuestas leyendo

las críticas del otro. Es así que se han cruzado una serie de artículos J2EE vs Microsoft.Net:

<http://www.theserverside.com/resources/article.jsp?l=J2EE-vs-DOTNET>, <http://www.objectwatch.com/FinalJ2EEandDotNet.doc>, es la respuesta de Microsoft.

[http://www.softwarereality.com/programming/component\\_oriented.jsp](http://www.softwarereality.com/programming/component_oriented.jsp)

Un análisis comparativo de los distintos elementos del Middleware CCM, EJB, COM+/MTS está disponible en

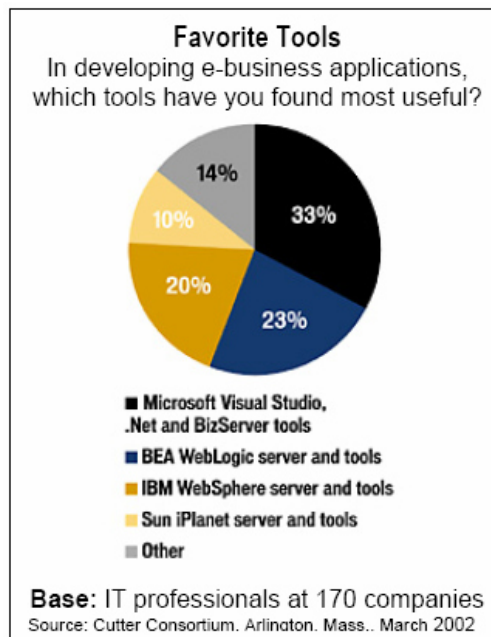
<http://my.execpc.com/~gopalan/misc/compare.html>

### **3.4.3 Conclusiones**

El cuadro siguiente muestra la distribución actual del mercado de middleware, en el se puede apreciar que mientras el 33% utilizan herramientas de Microsoft, el 53 utilizan productos basados en J2EE, mientras que del 14% restante no se dispone de información detallada sobre la plataforma que utilizan.

Si bien se muestra una tendencia a converger sobre Corba como elemento neutro para la interoperabilidad, fundamentalmente por el lado de Java y sus seguidores muchos de los cuales son además proveedores de servicios Corba, habiendo alcanzado un nivel de interoperación muy importante, no es posible que se unifiquen ambas propuestas, el “costo de oportunidad” o Time to Market”, siempre estará actuando en contra. Mientras la OMG es un consorcio sin fines de lucro, las empresas que componen el grupo J2EE se confrontan entre ellas y con Microsoft por el mercado. Para mejorar su competitividad ofrecen soluciones que no necesariamente están contempladas en el estándar.

El MDA abre una perspectiva interesante, no se apuesta más a generar un único estándar, sino que en base a una conceptualización estándar y un estándar de referencia (Corba), se puedan desarrollar sistemas con independencia de la plataforma final.



## 3.5 WEB SERVICES

### 3.5.1 Conclusiones

En octubre de 1994, se crea el World Wide Web Consortium (W3C), *“a fin de liderar el desarrollo de la Web a su máximo potencial, desarrollar protocolos comunes y asegurar su interoperabilidad, velar por la standardización mediante recomendaciones técnicas”*. Integrada en la actualidad por alrededor de 500 miembros, el W3C define los principios fundamentales de diseño de la Web: Interoperabilidad, Evolución y Descentralización.

En 1998 se adopta el “Extensible Markup Language” (XML) “El Extensible Markup Language” (XML) es la forma universal para documentos o datos estructurados sobre la Web.

En la dirección <http://www.w3.org/XML> en el documento “XML in 10 points”, la W3C expone brevemente las bases sobre las que se sustenta XML y las direcciones futuras. Hay dos factores principales que inciden en el desarrollo del XML: i) El “Business-to-

Business” (B2B) y “e-commerce” que requieren que las organizaciones que utilizan diferentes plataformas se comuniquen e intercambien datos. ii) Muchas instituciones necesitan poner a disposición de aplicaciones basadas en Web, información normalmente almacenada en sistemas de “backoffice”, como bases de datos y documentos <sup>1</sup>.

XML, no es un lenguaje de programación, XML es una familia de tecnologías: XML Protocol, XML Schema, XML Query , XLink, XPointer, XML Base, DOM, RDF, CSS XSL, XHTML, MathML, SMIL, SVG, XML Signature and Canonicalization son las más relevantes.

Una matriz completa del estado de los proyectos XML, se puede obtener en <http://www.w3.org/#technologies>.

El grupo XML Protocol Activity, fundado en setiembre de 2000 dirige las actividades basadas en un protocolo de mensajería fundado sobre XML, actualmente estudia lo relativo a Web Services. Este grupo se divide en tres subgrupos: Web Services Architecture Working Group. XML Protocol Working Group. Web Services Description Working Group los que se ocupan de los estándares vinculados a Web Services: “Web Services Description Language” (WSDL), “Simple Object Access Protocol” (SOAP) y “Universal Description Discovery and Integration” (UDDI). Una matriz de los protocolos homologados y sus versiones se puede encontrar en <http://www.w3.org/2000/03/29-XML-protocol-matrix>.

Existen una cantidad de propuestas complementarias para enriquecer las funcionalidades y posibilidades de esta forma de integración de sistemas. Entre ellas las más importantes son las vinculadas a seguridad: “Ws Security” (Web Services Security) propuesta por Microsoft e IBM [www.ibm.com/developerworks/library/ws-secure/](http://www.ibm.com/developerworks/library/ws-secure/), lenguajes de “workflow” como “Xlang” de Microsoft y “WSFL” de IBM, lenguajes

---

<sup>1</sup> María-Angeles Grado-Caffaro, Martín Grado-Caffaro, The challenges that Xml Faces. IEEE, Computer October 2001

para la coordinación como “Ws-Coordination”, para la especificación de transacciones “Ws-Transaction”, lenguajes para la descripción de proceso de negocios “BPEL4WS” (Bea, IBM y Microsoft) <http://dev2dev.bea.com/techtrack/BPEL4WSprint.jsp>.

La mayoría de estas propuestas han sido realizadas entre los meses de abril y agosto de 2002 y aún no están homologadas por la W3C. No obstante, hay productos comerciales que ya utilizan algunas de ellas como Xlang (que es un poco anterior y WSFL).

En Abril de 2002, se crea una nueva organización la “Web Services Interoperability organization” (WS-I) [www.ws-i.org](http://www.ws-i.org) que se define a si misma *“como una organización abierta destinada a promover la interoperabilidad de los Web Services, entre plataformas, sistemas operativos y lenguajes de programación diferentes”*. Esta organización trabaja con las industrias y organizaciones de estándares para satisfacer las necesidades de los usuarios proveyéndoles de ayudas, mejores prácticas y recursos para el desarrollo de soluciones basadas en Web Services.

La WS-I se ha iniciado con problemas de integración y algunos fabricantes importantes como Sun Microsystem, han quedado fuera del grupo de dirección. No se puede decir que ha sido un buen comienzo para una organización que pretende promover la interoperabilidad.

Pese a la cantidad de propuestas existentes y a la lucha entre las diferentes empresas por tomar la delantera, pero gracias a la necesidad de interoperar sobre Internet, es que los Web Services ofrecen un nivel de estandarización interesante: i) Protocolos de comunicación estándar. ii) Formato de representación estándar de los datos. iii) Lenguaje de descripción estándar. iv) Un mecanismo estándar de “discovery”.

*“La potencia de los Web Service, radica además de su gran interoperabilidad y extensibilidad gracias al uso de XML, en que ellos pueden ser combinados para proveer operaciones mas complejas. Varios programas proveyendo servicios simples pueden interactuar para permitir operaciones complejas.*

*A fin de poder obtener una automatización completa de este tipo de interacciones, la arquitectura de los Web Services necesita mayor entendimiento y el desarrollo de algunas nuevas tecnologías”.<sup>9</sup>*

Además de la W3C hay otros grupos trabajando sobre XML, hay muchas formas de utilizar el metalenguaje y su potencia permite que se trabaje en múltiples direcciones. Lamentablemente no todas están sincronizadas existen, tal vez, demasiadas propuestas basadas en XML que en lugar de acelerar el proceso de integración, pueden llegar a producir desconcierto.

### **3.5.2 ebXML**

En ese contexto la “Organization for de Advancement of Structured Information Standards” (Oasis) [www.oasis-open.org](http://www.oasis-open.org) y la “United Nations Centre for Trade Facilitation and Electronics Business” (UN/CEFACT) [www.uncefact.org](http://www.uncefact.org), han desarrollado y adoptado ebXML (Electronic-business XML) [www.ebxml.org](http://www.ebxml.org) .

En <http://www.ebxml.org/implementations/index.htm> se detallan las implementaciones existentes y en [www.ebxml.org/techspecs](http://www.ebxml.org/techspecs), se detallan las especificaciones técnicas conjuntamente con el estado de las propuestas.

El ebXML, está diseñado y orientado al e-business y se basa en la integración con otras propuestas para unificarlas. Si bien tanto el modelo como los instrumentos son adecuados para el comercio electrónico, tiene menos éxito comercial que los Web Services. Quizás en ese aspecto lo más significativo es que tanto Microsoft como Ibm, socios de ebXML, apoyan poco esta iniciativa, a la vez que intentan fortalecer los “web services” mediante propuestas conjuntas a la W3C. Propuestas que tienden a fortalecer los “web services” en aspectos que, en ebXML ya están encaminados, como por ejemplo seguridad y descripción de procesos de negocios. Un análisis interesante de este aspecto, que sin duda involucra al futuro de ebXML, se presenta en.

### 3.5.3 RosettaNet

Hay mas consorcios y organizaciones sin fines de lucro, que intentan aportar a la estandarización e interoperabilidad para el comercio electrónico y en particular para un problema que aparece tan complejo como necesario: la cadena de suministros “Supply-chain business processes”. RosettaNet (<http://www.rosettanet.org>) integrada por 400 miembros se define a si misma como: *“RosettaNet es un consorcio sin fines de lucro integrado por las principales empresas de tecnología de la información, componentes de electrónica y manufactura de semiconductores, que trabaja para crear e implementar a lo ancho de la industria, estándares abiertos de proceso de negocios para ebusiness. Esos estándares constituyen un lenguaje común para el e-business, alineando procesos entre socios de una cadena de suministros en un ambiente global.*

RosettaNet es una subsidiaria del “UniformCode Council” (Ucc). Este consorcio define una serie de estándares que se pueden encontrar en <http://www.rosettanet.org/standards>. El modelo se base en la definición de interfaces “Partner Interface Processes” (PIP) y de un “framework” el “RosettaNet Implementation Framework” (RNIF) en el cual se describen los PIPs para cada componente del proceso de la cadena de suministros “Supply Chain”<sup>1</sup>.

---

<sup>1</sup> Rajeev Kak, Director, High Tech Industry Marketing, i2 Technologies Dave Sotero, Senior Marketing Manager, RosettaNet; Implementing RosettaNet E-Business Standards for Greater Supply Chain Collaboration and Efficiency

## **Capítulo 4: DESCRIPCION DE LA SOLUCION TECNOLOGICA**

Este capítulo describe el escenario y la solución tecnológica adoptada para la integración de los sistemas de información heredados en una Institución pública educativa. Además se justifica por que se opto esta alternativa de solución.

Luego de hacer un análisis y estudio de las diferentes alternativas y técnicas para la interoperatividad de sistemas de información en el capitulo anterior, vemos que los Web Services prometen una mejor opción para la integración de Sistemas de información diversos en plataformas heterogéneas, y así posibilitan la automatización y conexión dinámica de procesos dentro de las organizaciones mediante Web Services, la composición de servicios Web ha experimentado un gran interés debido a la posibilidad de desarrollar nuevos servicios a través de la composición de servicios individuales.

Desafortunadamente dado que los servicios individuales fueron desarrollados usando diferentes propuestas y tecnologías, se requiere actualmente una cantidad considerable de programación y esfuerzo para componer los servicios de forma que sea posible obtener una aplicación libre de errores y en un tiempo de desarrollo prudencial. Estas limitaciones han provocado un gran esfuerzo en investigación y desarrollo enfocado al área de integración en servicios Web, para contribuir al desarrollo de estándares, herramientas, modelos y lenguajes que permitan efectuar el diseño y ejecución de servicios satisfactoriamente.

En este trabajo se analizaron los estándares, técnicas, modelos y lenguajes presentes en distintas plataformas para Integración de Sistemas en entornos Web propuestas por grupos de investigación relevantes.

El objetivo principal de esta investigación es proponer una estrategia que permita a los trabajadores de los diversos departamentos acceder a información y funcionalidades de los viejos sistemas heredados en una aplicación integradora y en un entorno Web.

Entre los aspectos a tener en cuenta, están las especificaciones de la interfaces que usarán los empleados, los cambios requeridos en cuanto a la arquitectura de la red de la organización, la lógica de negocios, el acceso a

datos, la plataforma para la que se desarrollo, el sistema operativo instalado en servidores y estaciones de trabajo, etc., y en especial, las limitaciones que impone el tener que integrarse, de un modo u otro, afecta a casi todo lo demás.

Además, no se han olvidado otros objetivos no menos importantes que el sistema debe cumplir como son: el rendimiento, la seguridad, la disponibilidad, la consistencia, la escalabilidad, la administración, el mantenimiento, etc., que también se ven afectados por la integración

Para el desarrollo de este trabajo primeramente se presenta la arquitectura genérica del Web Service, la cual permitirá definir los requerimientos mínimos para cubrir todo el ciclo de vida de un sistema de composición de servicios: descripción y modelado de evaluación del servicio compuesto y ejecución y control de la composición.

#### **4.1 Arquitectura diseño general**

En esta sección presentamos una arquitectura genérica para la Integración de Sistemas Heredados con Web Service. Esta estructura no pretende reflejar la utilización de un lenguaje particular, plataforma u algoritmo de Integración. El objetivo de la arquitectura es presentar los requerimientos mínimos necesarios para cubrir todo el ciclo de vida de un sistema de Integración. Para que se inicie el proceso de interoperatividad, un proveedor de servicio debe describir y publicar un servicio; por otra parte un solicitante del servicio debe efectuar una solicitud para que se inicie el proceso de acceso a funcionalidades del sistema integrador. El proceso de definición y composición de servicios (figura 4.2) puede soportar la traducción de un lenguaje de diseño a un lenguaje más formal, el cual es utilizado por el generador del proceso de composición (figura 4.3). Debido a que algunos servicios pueden tener una funcionalidad muy similar, es posible que se genere más de un modelo del servicio compuesto que cubra los requisitos.



**FIGURA. 4.1** Arquitectura de Diseño General de Sistemas Heredados usando *Web Services*

Fuente: Elaboración Propia

## 4.2 Motivación

En la actualidad, muchas organizaciones públicas se encuentran adaptando sus procesos de negocios hacia los nuevos escenarios económicos y tecnológicos. En ese sentido, la modernización e integración de sus sistemas de software son tareas indispensables para lograr ese objetivo. Un problema a solucionar es la presencia de los llamados Sistemas Heredados, como parte del portafolio de sistemas que las organizaciones poseen.

Una característica interesante de la tecnología de *Web Services*, es que permite que los sistemas puedan “exponer” sus servicios, utilizando la tecnología estándar de *Internet*, para cualquier otro tipo de sistema que lo necesite sin que sea necesaria su modificación.

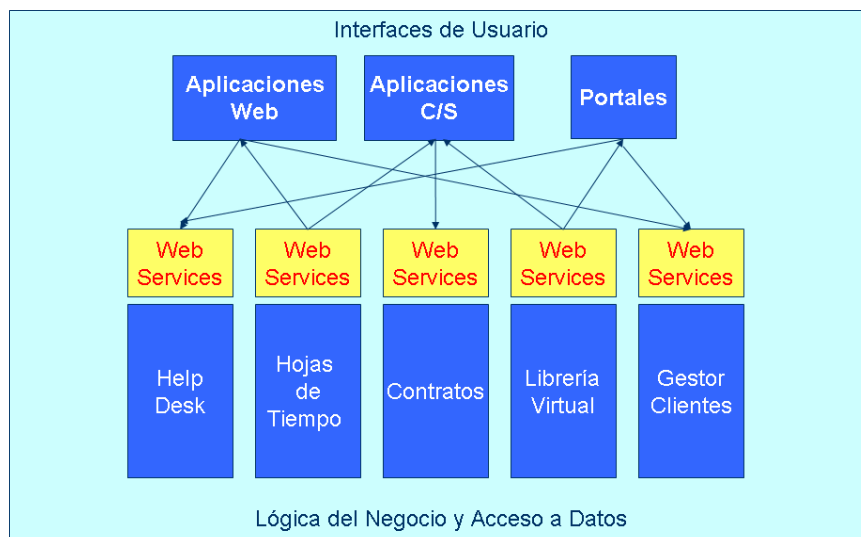


**FIGURA. 4.2** Integración de Sistemas Heredados usando *Web Services*

Fuente: Elaboración Propia

En la figura 4.2 se muestra un proyecto de integración, en la cual los desarrolladores crean interfaces para cada tipo de aplicación, para poder permitir una mejor comunicación entre sus sistemas. Debido a que cada interfase, para cada tipo de aplicación es autónoma, se integran grandes números de aplicaciones, y se genera una compleja red que requiere de niveles especializados para su desarrollo, incrementando los costos de integración y también incrementando la complejidad de los procesos de los negocios en las organizaciones. Si a este problema, se redujera el número de interfaces, se podría reducir la complejidad de integración de las aplicaciones. En ese sentido, los *Web Services*, permiten que los sistemas puedan “exponer” sus servicios a cualquier otro sistema, a través de una interfaz única que pueda ser usada por cualquier tipo de aplicación. Estos sistemas, pueden ser invocados directamente como una interfaz tradicional.

#### 4.3 Arquitectura conceptual de Integración de Sistemas Heredados Utilizando Web Services

A continuación, se muestra la arquitectura conceptual, propuesta para la utilización de la nueva estrategia de integración de sistemas utilizando *Web Services*.



**FIGURA. 4.3** Arquitectura Conceptual de Integración de Sistemas Heredados utilizando *Web Services*

Fuente: Elaboración Propia

La arquitectura conceptual mostrada, es la propuesta para la implementación de la nueva estrategia de integración de Sistemas Heredados utilizando *Web Services*. Esta estrategia permite reutilizar las funcionalidades del Sistema Heredado existente, permitiendo realizar llamadas desde el *Web Services* hacia una llamada de tipo convencional. El *Web Services*, se encargara de adaptar las funcionalidades para poder exponerlas en entornos de tipo Web, utilizando para ello un componente llamado *Wrapper*, permitiéndose a los servicios poder ser integrados y consumidos por nuevos tipos de sistemas.

A continuación, se muestra la funcionalidad de la estrategia de *Web Services*.

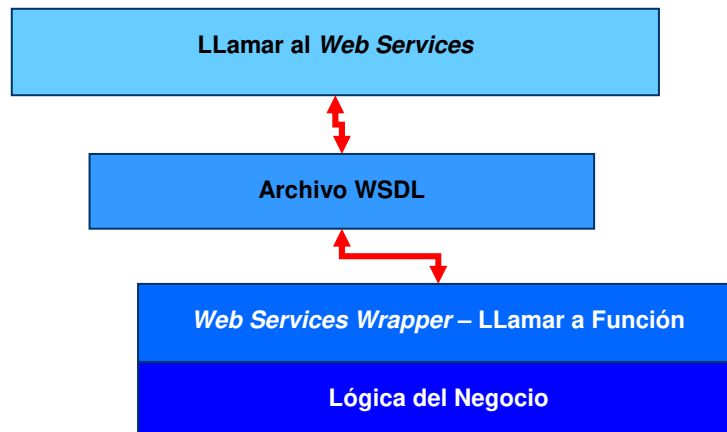


**FIGURA: 4.4** Arquitectura de Funcionalidad de la estrategia de integración de Sistemas Heredados usando *Web Services*

En la figura 4.4, se observa la funcionalidad de la estrategia de *Web Services*, el cual es iniciado desde el envío de una petición por parte del cliente para obtener un servicio, es decir llamar al *Web Services*. Esta petición se realiza por medio de mensajes a través del SOAP entre el cliente y el servidor del *Web Services*. Los servicios del *Web Services*, se encuentran dentro de un repositorio llamado UDDI, los cuales son definidos en interfaces especificados en WSDL dentro del estándar XML. El *Web Services (Wrapper)* se encargara de realizar llamadas a los servicios adaptados (Funcionalidades del Sistema Heredado), permitiendo la integración de los mismos, para poder ser consumidos por nuevos tipos de sistemas.

#### **4.4 Uso de la Técnica Wrapping para la Integración de los Sistemas Heredados**

Una de las técnicas para migrar Sistemas Heredados, a nuevos tipos de sistemas, es realizando llamadas a las funcionalidades del Sistema Heredado existente, mediante el uso del *Wrapper*. La idea de *wrapper*, las funcionalidades del Sistema Heredado para ser reutilizado, es un concepto que fue introducido por primera vez por W. Dietrich en 1989 y desde entonces se ha ido ampliando y difundiendo. Actualmente, la necesidad de incorporar a *Internet* servicios proporcionados por Sistemas Heredados, conjuntamente con la mayor difusión de Corba, J2EE, COM y .NET, ha impulsado notoriamente esta idea.

Mediante *wrapper*, es posible adaptar al Sistema Heredado, para que sea parte de una nueva generación de sistemas, sin los riesgos inherentes a la reescritura y los altos costos de nuevos desarrollos. Es evidente, que para que esto no introduzca problemas a futuro, el estado de salud del sistema debe ser “sano” o debe ser factible pasarlo a dicho estado. Otro de los aspectos importantes de esta técnica, es que permite adaptar partes de un sistema según las necesidades y las urgencias, manteniendo el resto inalterado.

El *Wrapping*, es una estrategia parecida a la de *Message Broker*, debido a que utiliza un controlador que coordina los movimientos de información entre los Sistemas Heredados a ser integrados mediante mensajes.

#### **4.5 Ventajas de esta estrategia de integración**

En este capítulo, se ha conocido la estrategia de *Web Services*, como nueva solución al problema de integración de Sistema

Heredados. Por ello, los Sistemas Heredados, incluyen aplicaciones creadas en distintas plataformas y escritas en distintos lenguajes. Para crear soluciones, es esencial integrar estos sistemas. La integración de Sistemas Heredados, creados en distintas plataformas es más fácil con el uso de *Web Services*.

Un *Web Services*, es un componente reutilizable, como un método, que puede ser utilizado por cualquier aplicación *Web* que se ejecute en *Internet*. Estas aplicaciones, se llaman aplicaciones cliente de *Web Services*. La aplicación que aloja al *Web Services* se llama proveedor de servicio *Web*.

En el capítulo cinco, se presenta la implementación de la estrategia de *Web Services*.

#### **4.6 Descripción de la Solución Segunda Estrategia**

Como solución al problema anteriormente descrito, se requiere de la utilización de la tecnología de *Web Services*, como nueva estrategia de integración de sistemas. El *Web Services*, gracias al uso de sus estándares (WSDL, UDDI, XML y SOAP), permite la adaptación de diferentes tipos aplicaciones (Ejemplo: funcionalidades del Sistema Heredado) hacia un entorno de servicios tipo *Internet*, para que estas a su vez puedan ser publicadas y consumidas por nuevos tipos de plataformas tecnológicas.

#### **4.7 Arquitectura Física de Integración de Sistemas Heredados utilizando *Web Services***

A continuación, se muestra la arquitectura física propuesta para la integración de Sistemas Heredados utilizando *Web Services*. (Ver figura 4.5).

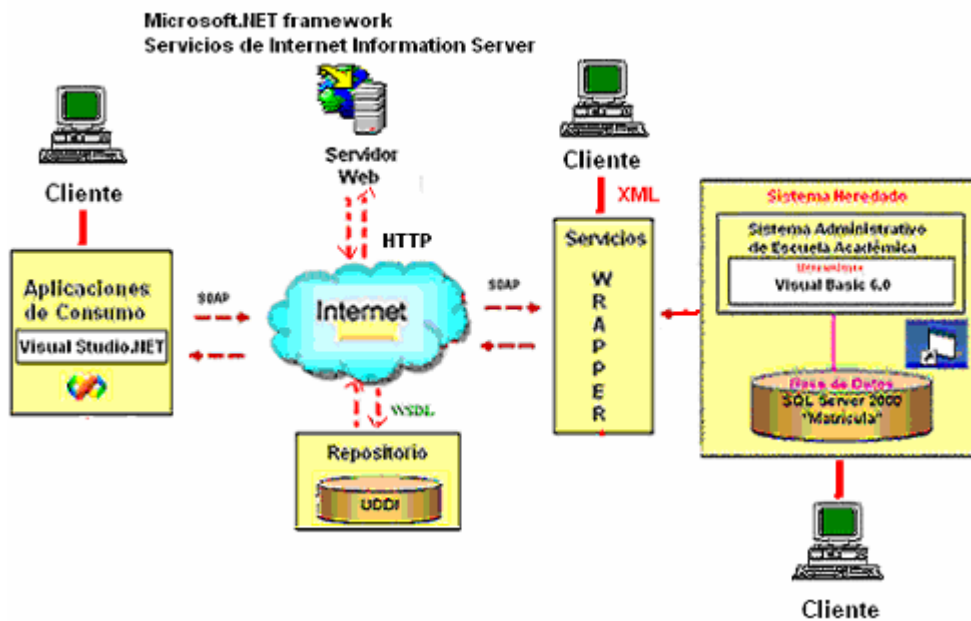


**Figura 4.5.** Arquitectura Física de Integración de Sistema Heredados Usando Web Services

La arquitectura física esta compuesto de la siguiente manera: (1) **Sistema Heredado**, es un Sistema Heterogéneo, tiene algunas funcionalidades (alumnos, carreras, docentes y cursos, matricular alumnos) para un Sistema Académico Administrativo. Este sistema esta desarrollado bajo la plataforma de *Visual Basic 6.0* y un servidor de Base de Datos en *SQL Server 2000* ("Matricula"), con sus tablas relacionales y Stored Procedure. (2) **Wrapper**, es un componente que permite extraer aquellos elementos importantes para el intercambio de información desde las funcionalidades del Sistema Heredado existente, este componente esta incluido dentro de la lógica del negocio del **Web Services**, la cual permite realizar llamadas desde un *Web Services* hacia llamadas de tipo convencional, permitiendo adaptar y exponer las funcionalidades (alumnos, docentes, cursos, y carreras, matricular alumnos), para luego poder ser integrados y consumidos en el desarrollo de nuevos sistemas. El *Web Services*, esta implementado bajo la plataforma tecnológica de *Microsoft .NET Framework* y cuyo servidor *Web*, trabaja con directorios virtuales gracias al uso del *Internet Information Server* (IIS). (3) **Aplicaciones de Consumo**, es la

implementación de una nueva aplicación, que permita consumir las funcionalidades adaptadas por el *Web Services*, a manera de consulta de servicios. Esta aplicación esta implementado en Visual C#.NET con interfaces de consumo.

## 4.8 Implementación de Integración de Sistemas Heredados utilizando Web Services

### 4.8.1 Objetivo de la Implementación

El **objetivo** de la presente tesis, es la de poder demostrar de manera sistematizada el uso de la nueva estrategia de ***Web Services***, como una solución al problema de integración de sistemas. Por lo cual, se pretende reutilizar las funcionalidades del Sistema Heredado existente (alumnos, carreras, docentes y cursos, matricular alumnos), realizando llamadas desde un *Web Services* hacia llamadas de tipo convencional, para poder adaptar las funcionalidades existentes hacia una nueva plataforma tecnología para su posterior consumo. (Ver figura 5.2).

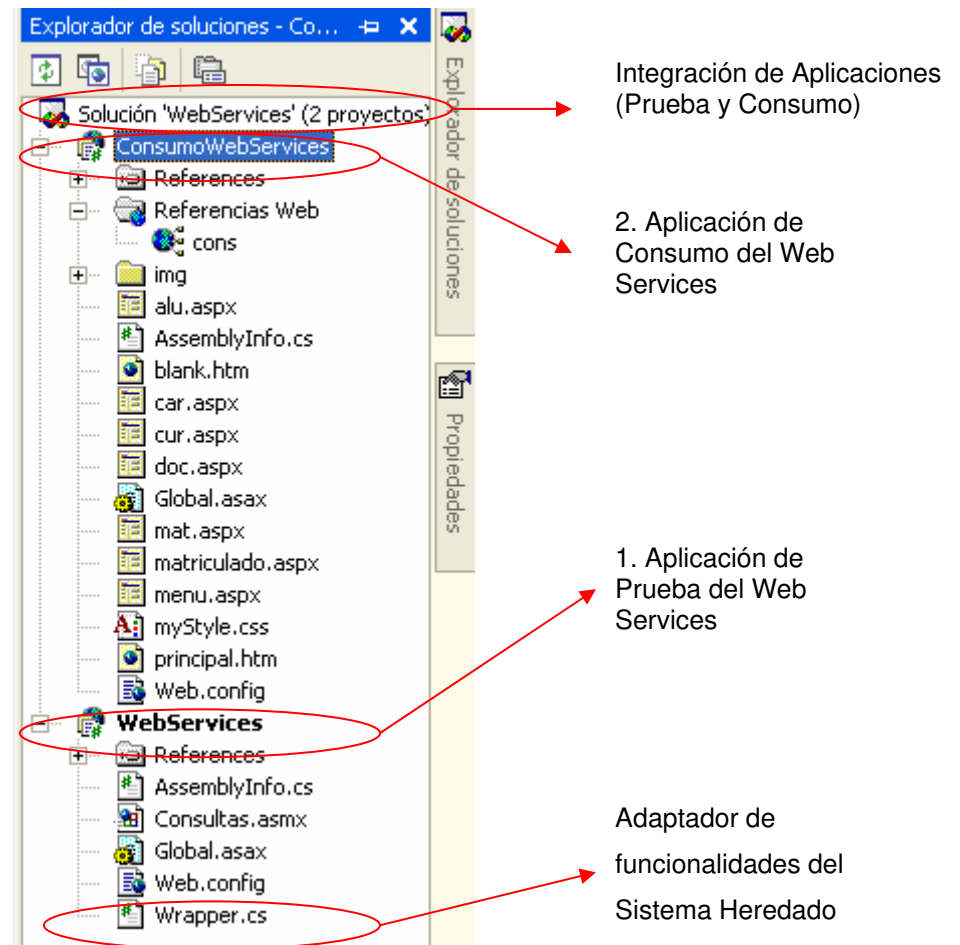


**Figura 4.6.** Integración de dos aplicaciones: Prueba (WebServices) y Consumo (ConsumoWebServices).

A continuación, se muestra el diagrama de flujo, con los procesos requeridos para la implementación de la estrategia de *Web Services*. (Ver figura 4.7).

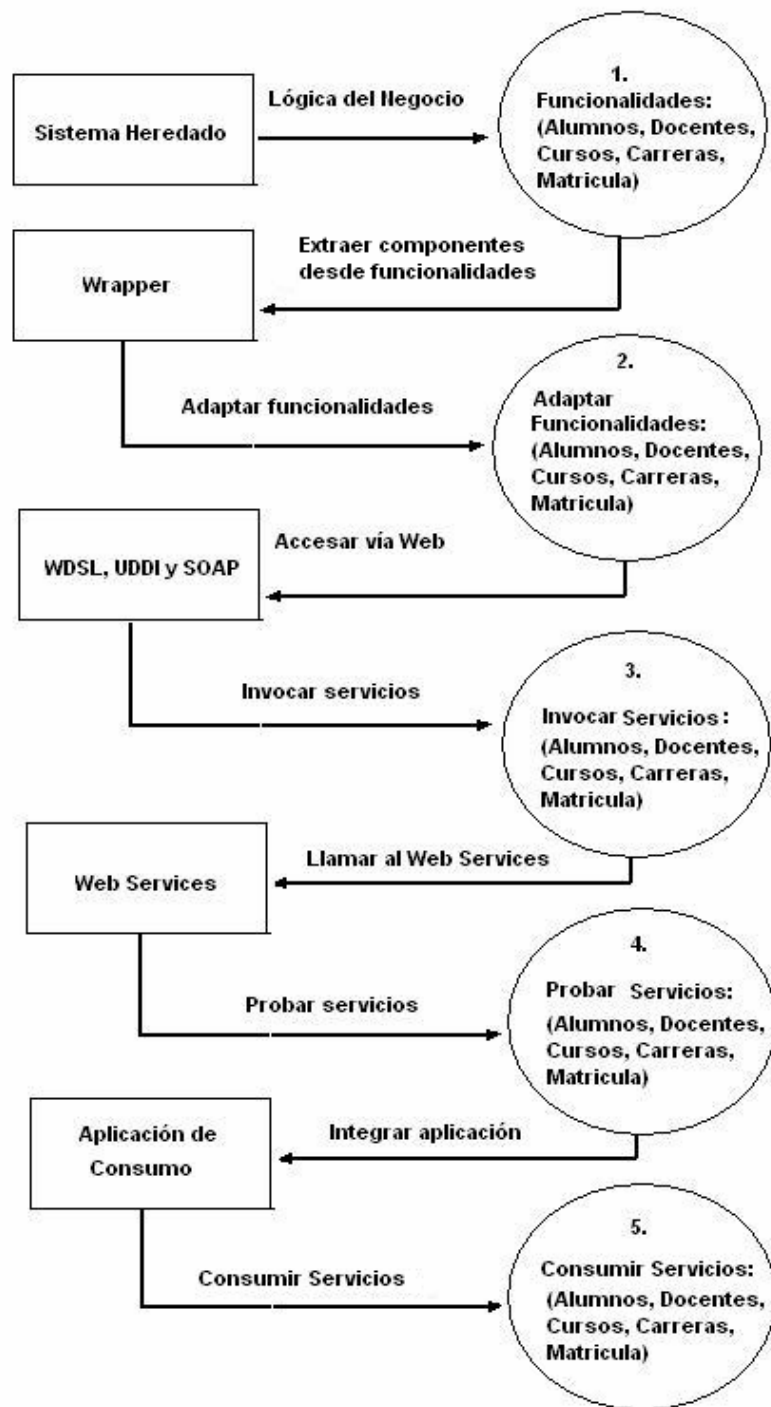


**Figura 4.7.** Diagrama de Flujo de la estrategia de Integración de sistemas Heredados utilizando Web Services

#### **4.8.2 Sistema Heredado**

Las interfaces, aplicaciones y servicios de Base de Datos de un Sistema Heredado, pueden ser considerados como componentes diferentes con interfaces bien definidas. Consisten en un conjunto de aplicaciones independientes, cada una de las cuales interactúa con los servicios de Base de Datos, y potencialmente con sus propias interfaces de usuario y de sistemas. Estas aplicaciones, son consideradas como aplicaciones Heterogéneas, ya que pueden estar creadas en distintas plataformas tecnológicas y escritos en distintos lenguajes de programación.

El Sistema Heredado, que se muestra en esta sección está desarrollado bajo la plataforma de Visual Basic 6.0 y un servidor de Base de Datos en SQL 2000. La aplicación cumple con las siguientes funcionalidades:

- Registrar matricula de alumnos.
- Realizar un mantenimiento de registros de: alumnos, docentes, carreras y cursos.
- Generar reportes de: alumnos, docentes, carreras y cursos.

A continuación, se muestra un diagrama de flujo con las funcionalidades del Sistema Heredado. (Ver figura 4.8).

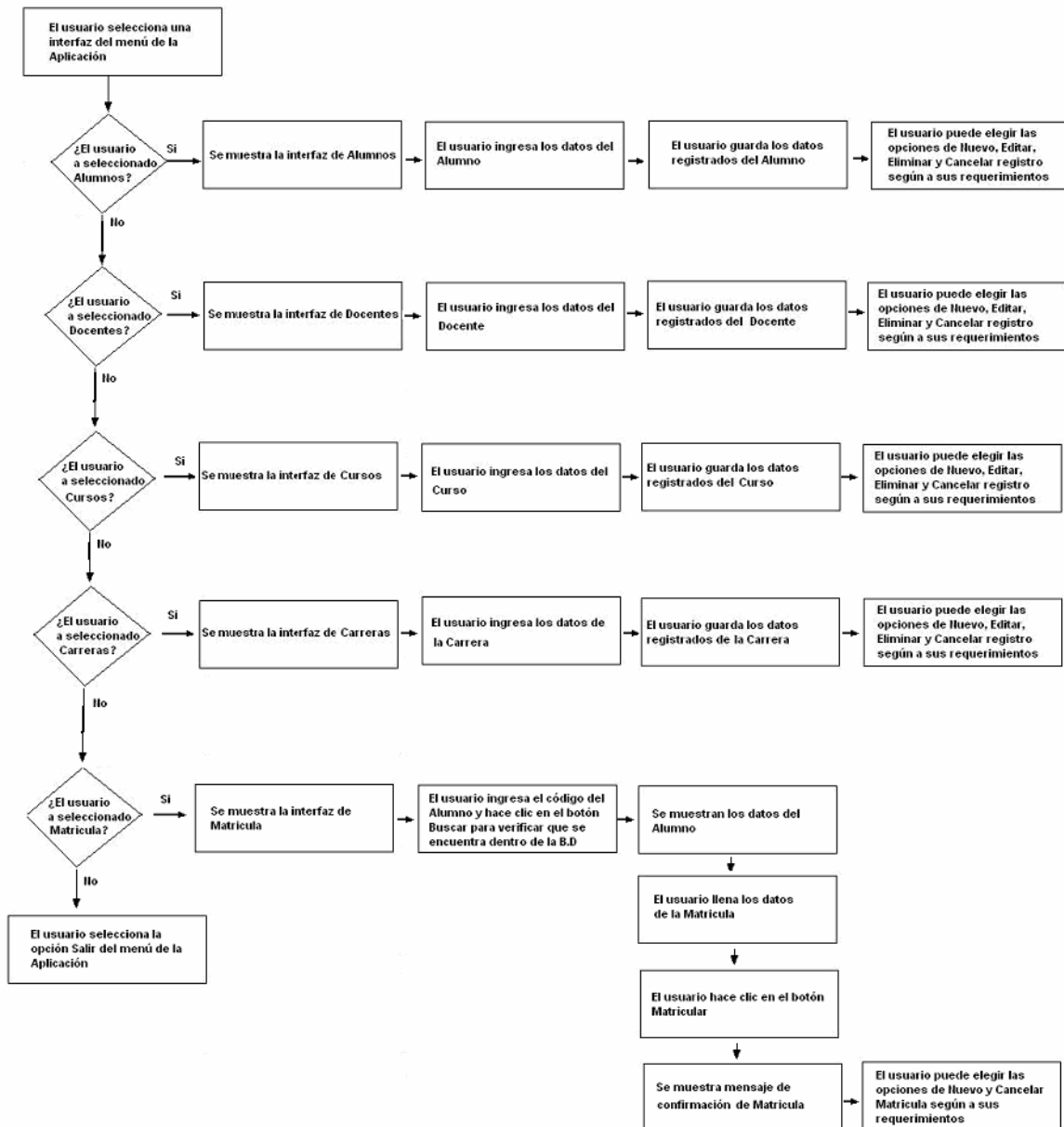


**Figura 4.8.** Diagrama de Flujo de funcionalidades del Sistema Heredado

Ya estando familiarizado con la importancia en cuanto a la funcionalidad del Sistema Heredado, el problema radica en poder reutilizar sus funcionalidades desde sus fuentes de datos, para que estas pudieran ser reutilizadas y expuestas en nuevas plataformas tecnológicas. Las interfaces del Sistema Heredado, están incluidas en el Anexo 2 (Página 108). A continuación, se muestra los componentes del Sistema Heredado. (Ver tabla 4.1):

|                                                                                                   |                                                                                                                                                    |
|---------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
|  SistemaHeredado | Ejecutable (SistemaHeredado.exe) desarrollado en Visual Basic 6.0, contiene las funcionalidades del Sistema Heredado.                              |
|  MATRICULA       | Data (Matricula) del Sistema Heredado, almacenada en un servidor de Base de Datos en SQL Server 2000, con tablas relacionales y stored procedures. |

**Tabla 4.1:** Componentes del Sistema Heredado.

Desde el punto de vista tecnológico el camino mas sencillo y para el cual existe una mayor cantidad de herramientas y trabajos realizados, es el acceder directamente a los datos del Sistema Heredado, para no perder la lógica del negocio del mismo.

A continuación, se presenta el diseño de la Base de Datos (Matricula) creada en SQL Server 2000, con sus respectivas tablas relacionales (Ver figura 4.9).

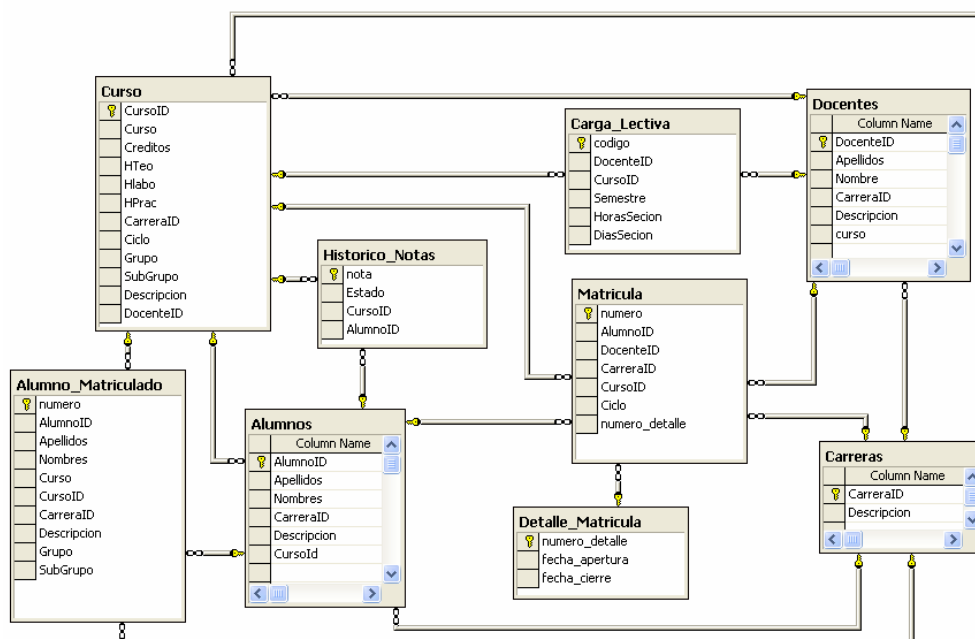


**Figura 4.9.** Diagrama de E-R de la Base de datos "Matricula"

#### 4.8.3 Web Services (Wrapper)

Los *Web Services* cuentan con la ventaja de que están soportados en forma nativa por cualquier plataforma, lo cual permite una solución de bajo costo y fácil de implementar. Por otro lado los *Web Services* tienen la desventaja que aún no permiten manejar transacciones entre plataformas, lo cual lleva a que la integración sea factible sólo para consultar información, no para la actualización. Una solución basada en *Web Services* permite la integración de cualquier plataforma.

La solución de implementar un *Web Services*, consiste en poder reutilizar las funcionalidades de un Sistema Heredado existente, extrayendo aquellos elementos importantes para la integración de la información, adaptando los mismos para permitir su exploración en el desarrollo de nuevos sistemas. Este prototipo, se encuentra implementado dentro del *Servidor Web* y bajo la plataforma de Visual Studio.NET, el cual gracias al uso del IIS, permite su navegación en entornos de tipo Internet (localhost).

El procedimiento de la implementación de la estrategia de *Web Services* es la siguiente:

1. Crear una clase *Wrapper* dentro de la lógica del negocio de la implementación del *Web Services*, la cual permitirá extraer aquellos elementos importantes dentro de las funcionalidades del Sistema Heredado existente, para poder adaptarlos, a través de sus fuentes de información de datos y poder llamarlos dentro de un “dataset” para establecer la conexión.

```
public class Wrapper //Creación de la clase  
Wrapper
```

```

{
private SqlDataAdapter da;
private DataSet ds = new DataSet();
private SqlConnection con;

...
...

```

Dentro del adaptador *Wrapper*, se procede a incluir los elementos a adaptar, es decir desde las fuentes de datos por medio de un dataset. El modificador de clase *new*, permitirá ocultar la clase heredada con el mismo nombre de la clase base.

```

...

...

public DataSet ConsultaAlumnos ()
//Adaptando funcionalidades de Alumnos
{
    da = new
SqlDataAdapter("spConsultaAlumno",
con);

//oculta la clase heredada con el mismo
nombre de la clase base.

...

...

}

...

...

}

```

2. Ya adaptados los elementos de las funcionalidades del Sistema Heredado, se procede a realizar llamadas desde

un *Web Services* hacia llamadas de tipo convencional, por medio de sus métodos, permitiendo que los mismos puedan ser expuestos por el *Web Services*, y poder ser reutilizados en el desarrollo de nuevos sistemas para su posterior consumo.

```
[WebMethod] // Llamada de Web Services a
llamada de tipo convencional.
public DataSet ConsultaAlumnos ()
{
    Wrapper w=new Wrapper();
    return w.ConsultaAlumnos();
}
```

3. La aplicación cliente que necesita acceder a las funcionalidades adaptadas que expone el *Web Services* necesita una forma de resolver la ubicación del servicio remoto. Se logra mediante un proceso llamado UDDI. El sitio *Web* en la cual se expondrán los servicios del *Web Services* es la siguiente:

<http://localhost/SistemaLegacy/Consultas.aspx>

4. Una vez que se ha resuelto el extremo de un *Web Services*, el cliente necesita suficiente información para interactuar adecuadamente con el mismo. La descripción de un *Web Services* implica meta datos estructurados sobre la interfaz que intenta utilizar la aplicación cliente así como documentación escrita sobre el *Web Services*, para ello describe sus servicios en un WSDL con un esquema estándar en XML.
5. Para el intercambio de datos, el cliente y el servidor tienen que estar de acuerdo en un mecanismo común de

codificación y formato de mensaje. El uso de un mecanismo estándar de codificar los datos asegura que los datos que codifica el cliente los interpretara correctamente el servidor, utilizando para ello un servicio de mensajería de petición y respuesta por medio de un protocolo estándar llamado SOAP.

6. Los datos que se transmiten entre el cliente y el servidor necesitan codificarse en un cuerpo de mensajes, para que estos puedan integrar aplicaciones, en múltiples lenguajes de programación y diversas plataformas tecnológicas, utilizando para ello un formato estándar XML, lográndose así la prueba de servicios del *Web Services*.
7. Una vez que se ha dado el formato al mensaje y se han serializado los datos en el cuerpo del mensaje, se deben transferir entre el cliente y el servidor utilizando como protocolo de transporte HTTP.
8. Ya habiendo definido y publicado los servicios del *Web Services* dentro de un Repositorio llamado UDDI, estos podrán ser encontrados por nuevas aplicaciones de consumo, para poder lograr esto, se hizo una referencia *Web* a las interfaces de prueba del *Web Services*.

A continuación, se muestra los componentes del *Web Services*. (Ver tabla 5.2):

|                                                                                                     |                                                                           |
|-----------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------|
|  AssemblyInfo.cs | Contiene los metadatos del ensamblado del proyecto.                       |
|  Global.asax     | Contiene el código para gestionar los eventos generados en la aplicación. |

|                                                                                                                                                                                                                                                                                        |                                                                                                                                                                                                                                                                                                                                                        |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  Web.config                                                                                                                                                                                           | Contiene información sobre los ajustes de los recursos ASP.NET.                                                                                                                                                                                                                                                                                        |
|  WebServices.csproj.webinfo                                                                                                                                                                           | Contiene información sobre la ubicación del proyecto en el servidor de desarrollo.                                                                                                                                                                                                                                                                     |
|  WebServices                                                                                                                                                                                          | Este archivo contiene los metadatos de la solución. Si el servidor de desarrollo es máquina local, este archivo existe en el servidor local.                                                                                                                                                                                                           |
|  References<br><ul style="list-style-type: none"> <li>* System</li> <li>* System.Data</li> <li>* System.Drawing</li> <li>* System.Web</li> <li>* System.Web.Services</li> <li>* System.XML</li> </ul> | La carpeta de Referentes, contiene referencias a los espacios de nombres empleados para desarrollar el servicio de <i>Web Services</i> . Por ejemplo, la carpeta Referentes, contiene los ficheros System, System.Data, System.Drawing, System.Web, System.Web.Services, y System.XML que contienen referencias a los respectivos espacios de nombres. |
|  Wrapper.cs                                                                                                                                                                                           | Clase que contiene el código <i>Wrapper</i> , para realizar llamadas desde un <i>Web Services</i> , esta clase permite adaptar las funcionalidades del Sistema Heredado.                                                                                                                                                                               |

**Tabla 4.2:** Componentes del *Web Services (Wrapper)*

## 4.9 Aplicaciones de Consumo

Gracias a la utilización del *Web Services*, en cuanto a la adaptación de las funcionalidades del Sistema Heredado, se ha logrado poder reutilizar las funcionalidades de la aplicación antigua y exponerlas de manera de consulta hacia una nueva plataforma tecnológica como son las Aplicaciones de Consumo, referenciando de manera *Web* las interfaces del adaptador *Web Services*, para que estas puedan ser expuestas para su posterior consumo. Esta aplicación está implementada en Visual C#.NET con interfaces de consumo.

A continuación, se muestra los componentes de la Aplicación de Consumo. (Ver tabla 4.3):

|                                                                                                        |                                                        |
|--------------------------------------------------------------------------------------------------------|--------------------------------------------------------|
|  ConsumoWebServices | Este archivo contiene los metadatos de la solución. Si |
|--------------------------------------------------------------------------------------------------------|--------------------------------------------------------|

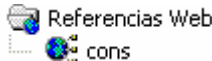
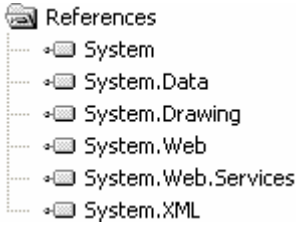
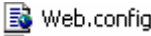
|                                                                                     |                                                                                                                                                                                                                                                                                                                                                        |
|-------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                                                     | el servidor de desarrollo es máquina local, este archivo existe en el servidor local.                                                                                                                                                                                                                                                                  |
|    | Esta carpeta contiene una referencia Web para exponer servicios (llama a los servicios de la aplicación Cliente del Web Services) desde interfaces de otros tipos de aplicaciones construidas en cualquier lenguaje de programación .NET de manera integrada                                                                                           |
|    | La carpeta de Referentes, contiene referencias a los espacios de nombres empleados para desarrollar el servicio de <i>Web Services</i> . Por ejemplo, la carpeta Referentes, contiene los ficheros System, System.Data, System.Drawing, System.Web, System.Web.Services, y System.XML que contienen referencias a los respectivos espacios de nombres. |
|    | Contiene los metadatos del ensamblado del proyecto.                                                                                                                                                                                                                                                                                                    |
|    | Contiene el código para gestionar los eventos generados en la aplicación.                                                                                                                                                                                                                                                                              |
|   | Contiene información sobre los ajustes de los recursos ASP.NET.                                                                                                                                                                                                                                                                                        |
|  | Contiene información sobre la ubicación del proyecto en el servidor de desarrollo.                                                                                                                                                                                                                                                                     |

**Tabla 4.3:** Componentes de la Aplicación de Consumo

## 4.10 Resumen del Capítulo

En este capítulo, se ha demostrado de manera sistematizada la aplicabilidad del uso de la nueva estrategia de *Web Services*, para el problema de integración de Sistema Heredados. Esta estrategia demuestra como poder integrar un Sistema Heredado existente y adaptarlo hacia una nueva plataforma tecnológica como es el *Web Services*, para su posterior consumo. También, se demuestra la utilización del servicio *Wrapper* incluido en la lógica del negocio del servicio de *Web Services*.

En el Capítulo 5, se describen las conclusiones, recomendaciones y trabajos futuros de la presente tesis.

## Capítulo 5: Conclusiones y trabajos futuros

### 5.1 Conclusiones

- La aparición de nuevas tecnologías, obligan a las organizaciones a integrar sus Sistemas Heredados hacia nuevas plataformas tecnológicas, adaptando las funcionalidades de sus sistemas, para poder ser reutilizados en el desarrollo de nuevos sistemas.
- Las Instituciones Públicas, en especial las educativas, cuentan en la actualidad con información muy valiosa en sistemas legados que son utilizado como fuente de consulta. Es de mucha importancia acceder a esta información histórica e información de los sistemas modernos desde una única interfaz integradora, esto se consigue a través de un Web Service Integrador.
- Existen estrategias de integración, que apuntan al reemplazo total (rediseño) o gradual (migración) de estos sistemas. En muchos casos, estos esfuerzos han fracasado dado que los nuevos sistemas no tenían las mismas funcionalidades de los Sistemas Heredados, debido principalmente a que la documentación era muy pobre. Una estrategia más simple, es la de dotar de un nuevo visual al sistema, es decir, desarrollar una especie de “envoltorio” (wrapping) que permita acceder a sus funcionalidades.
- Con el advenimiento del concepto de Reuso de Software y dado que los Sistemas Heredados poseen funcionalidades que han demostrado su confiabilidad a lo largo de los años, sería interesante “exponer” estas para que puedan ser utilizadas en la construcción de otros sistemas. En ese sentido, la aparición de nuevas tecnologías computacionales, como por ejemplo Web Services, permite que estas funcionales puedan ser reutilizadas en el desarrollo de nuevos sistemas

- Con el desarrollo de la presente investigación, se ha logrado obtener como resultados: i) Extraer aquellos elementos utilizados por las funcionalidades del Sistema Heredado existente, por medio de un componente incluido dentro de la lógica del Web Services llamado Wrapper, para poder realizar llamadas desde un Web Services hacia llamadas de tipo convencional, permitiendo la adaptación de los mismos. ii) Utilizando los estándares (WSDL, UDDI, XML y SOAP) se realizó la invocación de las pruebas de publicación de los servicios adaptados por el Web Services, permitiendo que los elementos adaptados puedan ser expuestos a través de la Web. iii) Con la utilización del XML, se procedió a reutilizar e integrar los servicios adaptados, para poder ser consumidos en el desarrollo de nuevos sistemas de manera de consultas.
  
- La experiencia recogida en integrar Sistemas Heredados, y la poca bibliografía existente en la materia, han impulsado al desarrollo de esta investigación, como una nueva visión de conjunto a esta problemática. Se entiende, que este enfoque constituye un aporte interesante a la comunidad informática, ya que presenta elementos aplicables tanto para la integración de sistemas, como para reconstruir componentes a partir de Sistemas Heredados.
  
- Cabe mencionar que este problema no sólo se percibe en Entidades públicas educativas como es el caso de estudio de esta tesis, si no que la podemos observar en muchas instituciones tanto públicas como privadas, este trabajo de investigación pretende mostrar el camino a la solución de este problema y eliminar de esta forma islas de información existentes dentro de las organizaciones.

- Se adoptó Web Services como solución por ser la opción tecnológica más prometedora en lo referente a integración de sistemas, gracias a los estándares y protocolos que permiten la interoperatividad de sistemas desarrollados en diferentes plataformas y que serían imposibles de contemplar bajo rutinas de librerías.
  
- Los estudios sobre las técnicas de ingeniería inversa y reingeniería de software, aportaron elementos conceptuales y modelos concretos para la transformación de programas integrantes de sistemas legados e ilustraron sobre la factibilidad técnica de estas adaptaciones, mostrando casos exitosos.
  
- A partir de la identificación de un conjunto de problemas, una propuesta de solución y del análisis del estado del arte de las tecnologías relacionadas, se han desarrollado los estudios que se detallan a continuación:
  - i. Identificación de los elementos que inciden en el proceso de integración de sistemas legados. Se identificaron una serie de elementos que deben ser tenidos en cuenta al integrar sistemas en un entorno Web. Sobre la base de la identificación de dichos problemas, es que se definieron muchos de los principios en los que se fundamenta este trabajo, los cuales se detallan en la Introducción y fueron considerados a lo largo de esta tesis.
  - ii. Descripción de procedimientos para adaptar programas o piezas de software, con un enfoque orientado a la interoperabilidad. Para ello se cumplieron las siguientes actividades:
    - a. Se ha analizado el impacto que los distintos elementos que hacen a la heterogeneidad, tienen sobre la capacidad potencial de interoperar.

- b. Se ha realizado una clasificación de sistemas legados extendiendo la ya existente, realizada por Brodie y Stonebreaker, a distintas plataformas (la clasificación referida, sólo se ocupa de entornos “mainframe”).
  - c. Se han propuesto patrones de software para la adaptación de sistemas legados a partir de sus interfaces de usuario o de programa, ejemplificando sobre algunos casos que requieren adaptación de caja gris.
  - d. Se realizó un relevamiento de productos comerciales, que ofrecen la capacidad de adaptar programas a partir de sus interfaces, cuyo resultado se presenta en un cuadro de productos.
- iii. Definición de una estrategia para el desarrollo de un sistema de información Integrado por Web Services. El desarrollo de estos tipos de sistema, así como el desarrollo de una macro operación, requiere de procedimientos específicos, adicionales a los definidos tradicionalmente en la ingeniería de software. Los estudios realizados se concretan en una metodología, aplicable tanto para la construcción como para el mantenimiento Web Service contempla:
  - a. Estudio de factibilidad.
  - b. Metodologías para análisis de macro Operaciones
  - c. Detectar los componentes a ser construidos a partir de funcionalidades de los sistemas legados.
  - d. Documentar las macro operaciones, las operaciones atómicas y los requerimientos de adaptación de los programas.
  - e. Establecer recomendaciones sobre: Testing, administración de versiones, constitución de equipos de trabajo y de dirección de Web services.

## 5.2 Trabajos Futuros

En todo trabajo de este tipo, van surgiendo una serie de elementos que no se pueden incluir en el mismo, algunos de ellos, fueron mencionados en texto, trabajarlos constituye un compromiso con este proyecto. Hay otros aspectos vinculados al tema, que surgen como consecuencia de los estudios realizados, que no deberían quedar sin profundizar.

Uno de los grandes temas, por su importancia y complejidad, es la gestión de transacciones en ambientes integrados de sistemas heterogéneos, en el cual se da una mezcla de aplicaciones con automatismos para el manejo transaccional y en el otro extremo, aplicaciones que no implementan ningún tipo de función que permita deshacer operaciones o restaurar la información, en forma dinámica, al estado anterior a la ocurrencia de una falla. Hay investigaciones desarrolladas en esta temática, hace falta un trabajo de recopilación y estudio de las mismas a fin de realizar una propuesta que permita implementar instrumentos que proporcionen esta funcionalidad a un Web Services Integrador.

Los dos trabajos mencionados pueden realizarse exclusivamente en el campo teórico, pero sería sumamente importante contar con una implementación del modelo presentado. Esto proporcionará un campo de experimentación mucho más rico y permitirá la búsqueda de un proyecto piloto que le de la solidez, que el trabajo de campo aporta a la investigación académica. En consecuencia, implementar una versión de orquestador (cliente y servidor) con un intérprete o compilador para el lenguaje propuesto, así como los repositorios de macro operaciones y servicios integrados con los demás elementos que conforman el modelo propuesto, es otro de los trabajos a realizar. Este, por su naturaleza requiere, de un equipo de trabajo, un Laboratorio de Integración de Sistemas debería ser el agente que impulsara este emprendimiento.

La metodología de trabajo también requiere de evaluación, crítica y mejoras. En primera instancia es necesario recopilar los estudios de las implementaciones prácticas de integración, realizadas con anterioridad a este trabajo. En segundo lugar, impulsar su aplicación para conseguir una masa crítica que permita evaluar y corregir. Esta segunda parte, requiere

nuevamente de la colaboración de equipos de trabajo, en los que se requiere la colaboración entre el ambiente universitario, el profesional y empresas en donde aplicarlo.

Los puntos que siguen, responden al segundo grupo de trabajos que se mencionó inicialmente y apuntan a darle generalidad al trabajo, pero requieren de los anteriores para que tengan sentido.

En este trabajo se mostraron patrones para realizar adaptadores de software, es posible instrumentar un “framework” de adaptador que actúe como caja de herramientas para construir los adaptadores específicos que sean requeridos. Este “framework” encapsulará todas las funciones relativas a interoperabilidad así como a automatismos para la simulación de operadores, dejando al programador la tarea de modificar el programa legado. Algunos estudios vinculados a esto, se están realizando a nivel de trabajo de grado por parte de un grupo de estudiantes en el ámbito del Laboratorio de Integración de Sistemas.

En este trabajo se puso énfasis en el reuso, en primer lugar del código de los sistemas legados para dar soporte a procesos de negocios. En segundo lugar, para reutilizar las funciones exportadas no solamente a nivel de Web Services, sino que también desde otros sistemas. Sin embargo no se indicó como realizar eso, es interesante documentar la forma de utilizar tanto las macro operaciones como las operaciones atómicas con otros fines. Ej. Web Services, Workflow, modernización de interfaces de usuario, etc.

Finalmente, presenta interés estudiar e implementar la generación automática de código para la coreografía de una macro operación, según la gramática propuesta, a partir del diagrama de actividad UML que representa el proceso de negocios.

## REFERENCIAS BIBLIOGRÁFICAS

- [Millan03] Millan José, Adaptative Brain Interfaces, Communications of the ACM, v. 46, n. 3, 2003, pp. 74-80.
- [Zhai03] Zhai Shumin, What's in the eyes for attentive inputs, Communications of the ACM, v. 46, n. 3, 2003, pp. 34-39.
- [Maglio+03] Maglio Paul y Campbell Christopher, Attentive agents, Communications of the ACM, v. 46, n. 3, 2003, pp. 47-51.
- Brodie, M., Stonebraker, M., "Migrating Legacy Systems : Gateways, Interfaces and the Incremental Approach", Morgan Kaufmann, San Francisco, 1995.
- UDDI Version 3.0, Published Specification, 19, july 2002 [http://uddi.org/pubs/uddi\\_v3.htm](http://uddi.org/pubs/uddi_v3.htm).
- Mecella M, Batini C. "Cooperation of Heterogeneous Legacy Information Systems: A Methodological Framework". Proceeding of the 4th International Enterprise Distributed Object Conference (Edoc 2000), Makuhari, Japan, 2000. pp 3.
- Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley Longman, 1995, pages. 139 –150.
- Cristina Cifuentes, "Reverse Engineering and Reengineering Newsletter" IEEE's Technical Council on Software Engineering TCSE Reverse Engineering and Reengineering (RER) Committee, September 2001.
- Software Engineering Institute ,Carnegie Mellon University Reengineering Center "Perspectives on Legacy System Reengineering", 1995.
- Ignacio García, Macario Polo, Francisco Ruiz, Mario Piattini, "Servicios Web" Universidad de Castilla-La Mancha, España. (Enero, 2005). Pag 1-200. serviciosweb.pdf.

- Raul Ruggia, Jorge Besil, Carla Pais, Dario Sande. "Interoperabilidad entre Servidores de Aplicaciones Heterogeneos" (Julio, 2003) Pag 1-12. CITA03\_InteropServAplic(Final).pdf
- Jorge Abin de Maria. "Integración de Aplicaciones Encapsuladas para el Desarrollo de Sistemas de Información Cooperativos". Instituto de Computación. Facultad de Ingeniería. Universidad de la Republica. (Noviembre, 2002). Pág. 1-258. TesisMaestria.pdf.
- E. Nesime Tatbul, "the Use of XML in Software Engineering" (Mayo, 2000) Pag 1-8. tatbul00use.pdf.
- Libro: Michael Morrison "XML al descubierto".
- Peter Eles, Oliver Sims, "Building Business Objects". John Wiley & Sons, Usa 1998.
- Innovation, Software, and Reverse Engineering: Technological and Legal Issues "Reverse Engineering and Reengineering Newsletter" IEEE's Technical Council on Software Engineering TCSE Reverse Engineering and Reengineering (RER) Committee, September 2001.
- Sneed Harry M., "Wrapping Legacy COBOL Programs behind an XML-Interface" Proceedings of the Eighth Working Conference On Reverse Engineering (WCRE.01) 0-7695-1303-4/02 © 2002 \_ IEEE.
- Mowbray, T. Zahari, R.: "The Essential CORBA", John Wiley & Sons, New York, 1994, p. 231-265.
- Sneed Harry M., Rudolf Majnar, "A Case Study in Software Wrapping", International Conference on Software Maintenance pp. 86 Date: March 1998.
- Robert Shapiro, A Comparison of XPD, BPML and BPEL4WS - Cape Visions, <http://xml.coverpages.org/Shapiro-XPDL.pdf> , Published by ebPML.org. 'Rough Draft' version 1.4, August 27, 2002. 17 pages.
- Andrew Watson, Model Driven Architecture -An Executive Overview, Object Management Group Vice President and Technical Director. - Datamation , The engineering data newsletter May. 2002.

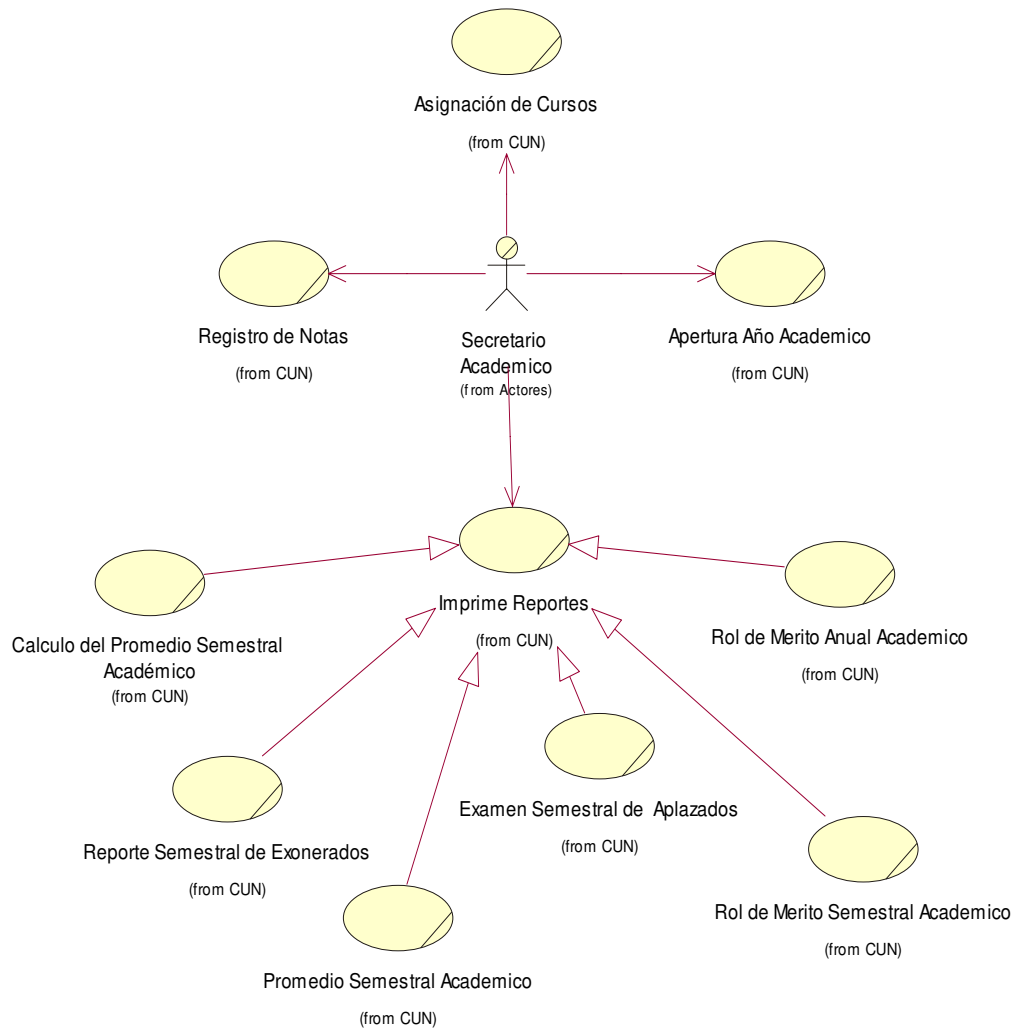


## **ANEXO 1**

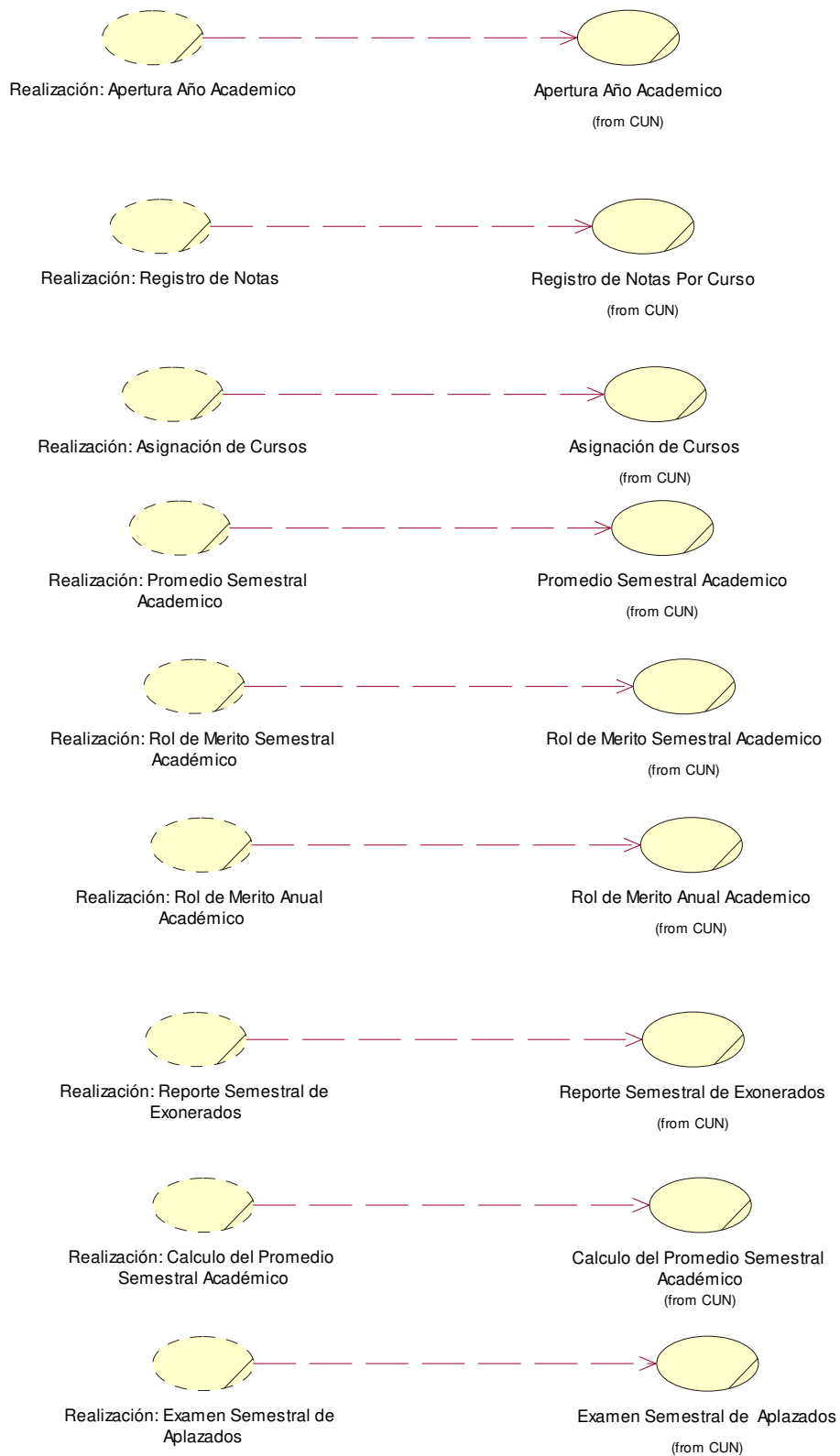
# **MODELADO DE LA IMPLEMENTACIÓN DE INTEGRACION DE SISTEMAS HEREDADOS UTILIZANDO WEB SERVICES**

## Modelo del Negocio

### a) Vista de Casos de Uso del Negocio Del Sistema Heredado

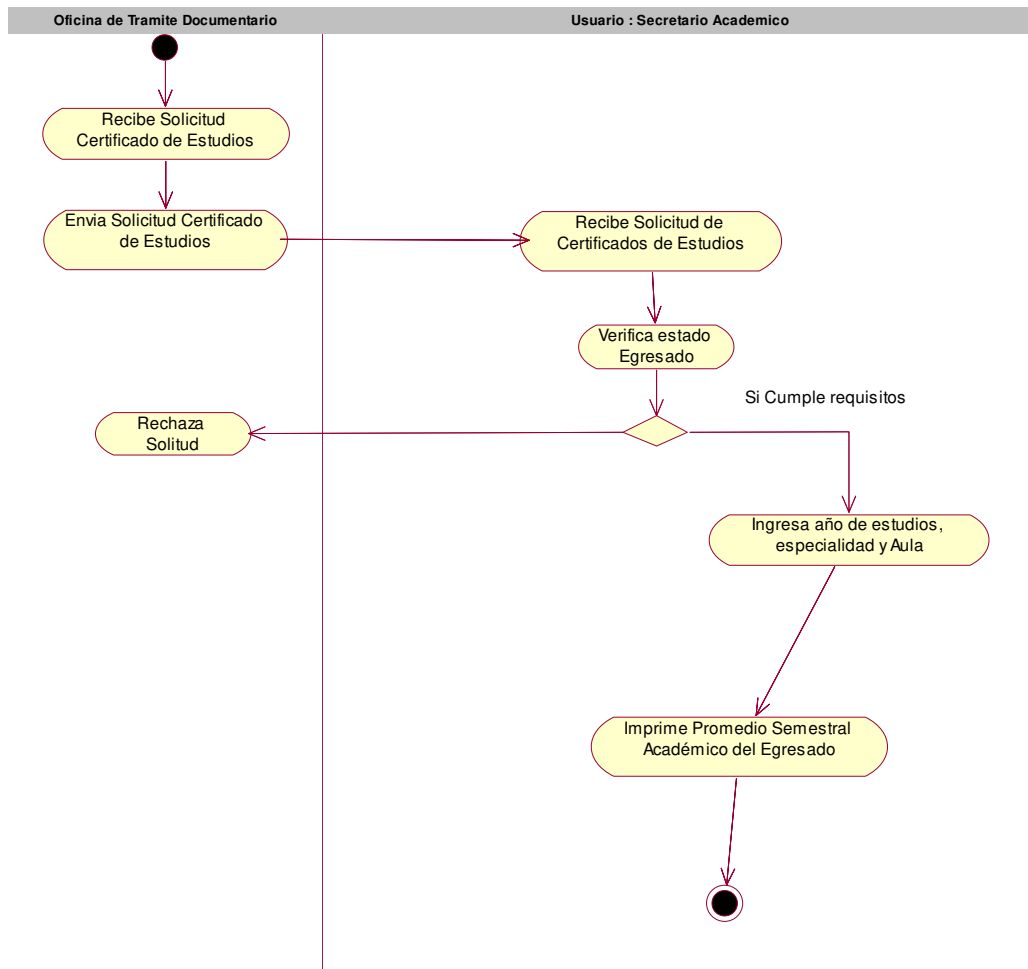


## b) Vista de Realizaciones de Casos del Negocio



c) Vista de Diagrama de Actividades de Casos de Uso del Negocio

a. Promedio Semestral Académico



**d) Especificaciones de Casos de Uso del Negocio**

**a. Realizar Apertura del Año Académico**

Este caso de uso, permite al usuario autorizado programar el año lectivo, Malla Curricular y formulas de evaluación.

**b. Realizar Registros de Notas**

Este caso de uso, permite al usuario autorizado registrar las notas obtenidas por el alumno de todas sus evaluaciones.

**c. Asignación de Cursos**

Este caso de uso, permite al usuario autorizado asignar el docente a un determinado curso.

**d. Promedio Semestral Académico**

Este caso de uso, permite el cálculo del promedio semestral académico, en función a las notas ingresadas.

**e. Rol de Merito Semestral Académico**

Este caso de Uso permite que la obtener el cuadro de merito en función a los promedios semestrales obtenidos.

**f. Rol de Merito Anual académico**

Este caso de uso, permite que la Secretaría académica consolide el rol de mérito anual académico.

**g. Reporte Semestral de Exonerados**

Este caso de Uso, permite obtener un reporte de asignación de becas a loa alumnos, tanto por su rendimiento académico o condición económica

## Modelo de Análisis

### 1. Visión del Sistema

#### 1.1. Posicionamiento

##### 1.1.1. Problema

|                            |                                                                                                                                                                                                                                       |
|----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| El problema de             | De la existencia de un Sistema Heredado, para administrar procesos académicos (alumnos, docentes, cursos, Notas, carreras, matricula alumnos).                                                                                        |
| Que afecta                 | Al personal administrativo, encargado de los procesos de generación de Certificados de estudios.                                                                                                                                      |
| y su impacto               | Necesidad de migrar y reutilizar las funcionalidades de la aplicación existente (alumnos, docentes, cursos, Notas, carreras, matricula alumnos), hacia nuevos sistemas tecnológicos.                                                  |
| Una buena solución debería | Utilizar la tecnología de <i>Web Services</i> , para reutilizar las funcionalidades del Sistema Heredado (alumnos, Notas, docentes, cursos, carreras), integrando aplicaciones y adaptándolos, para ser expuestos en nuevos sistemas. |

### 1.1.2. Posicionamiento del Producto

|                          |                                                                                                                                                                                                |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Para                     | Para el personal administrativo, encargado de los procesos académicos en las instituciones.                                                                                                    |
| Que necesitan            | Brindar un servicio de consulta de información (alumnos, docentes, cursos, notas), para gestiones administrativas realizadas.                                                                  |
| El (nombre del producto) | Servicios <i>Web</i> para Sistemas Académicos Administrativos.                                                                                                                                 |
| Que                      | Permitirá por medio de estándares de <i>Internet</i> poder realizar consultas de servicios (alumnos, docentes, cursos, Notas).                                                                 |
| A diferencia             | De la aplicación cliente/servidor actual (Sistema Heredado).                                                                                                                                   |
| Nuestro Producto         | Permitirá adaptar las funcionalidades existentes del Sistema Heredado (alumnos, docentes, cursos, carreras, Notas), para poder ser invocadas y reutilizadas en el servicio de nuevos sistemas. |

### 1.1.3. Descripciones de los Roles

#### 1.1.3.1. Resumen

| Nombre               | Descripción                                                                                                                    | Responsabilidades                                                                                                     |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------|
| Director             | Encargado de administrar la Escuela Nacional de Marina mercante                                                                | Encargado de aprobar los tramites y gestiones administrativos de los estudiantes, así como también para los docentes. |
| Secretaría Académica | Encargado de la atención de los estudiantes y egresados, como también formar parte de los procesos académicos que se realicen. | Asegurar que el sistema cumpla con las necesidades y las expectativas requeridas.                                     |

#### 1.1.4. Producto

##### 1.1.4.1. Necesidades y Características

| Necesidad                                                                                                                             | Prioridad | Característica                                                                                                                                                                  | Versión del Sistema |
|---------------------------------------------------------------------------------------------------------------------------------------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------|
| Director:<br><br>Información actualizada sobre los procesos administrativos (alumnos, docentes, cursos, carreras, matricula alumnos). | Alta      | El servicio de <i>Web Services</i> proporcionará una consulta de información (alumnos, docentes, cursos, Notas, carreras, matricula alumnos).                                   | 1                   |
| Administrativo:<br><br>Una reducción significativa del tiempo de acceso a las consultas realizadas.                                   | Alta      | El servicio de <i>Web Services</i> permitirá de manera ágil acceder a los servicios de consulta de información (alumnos, docentes, cursos, Notas, carreras, matricula alumnos). | 1                   |

### **1.1.5. Otros requerimientos del producto**

#### **1. Interfaz de Usuario**

Todas las interfaces, deberán ser expuestas en estándares de *Internet*, para los servicios proporcionados por el *Web Services*.

#### **2. Confiabilidad**

El sistema deberá de manejarse a través de la *Internet* de manera integrada.

#### **3. De rendimiento**

El tiempo de consulta de información de los servicios del *Web Services* para las gestiones administrativas, no deberá exceder los 5 minutos.

#### **4. Licencia e instalación**

Esto se realizara de acuerdo a las políticas de la institución.

## **2. Especificación de Requerimientos de Software**

### **1. Requerimientos Funcionales**

- Sistemas Operativos: Windows XP.
- Base de datos en SQL Server 2000.
- La aplicación de Sistema Heredado será en FoxPro for DOS.

### **2. Requerimientos No Funcionales**

#### **• Funcionalidad**

El sistema permitirá reutilizar las funcionalidades existentes del Sistema Heredado (alumnos, docentes, cursos, Notas, carreras, matricula alumnos), para ser invocadas utilizando los servicios del *Web Services*, y estos poder ser expuestos a nuevos sistemas.

#### **• Usabilidad**

El sistema tendrá la capacidad de ser entendido, aprendido, usado y resultar atractivo para el usuario, cuando sea utilizado en condiciones específicas.

- **Fiabilidad**

El sistema tendrá la capacidad de poder mantener un nivel especificado de desempeño cuando sea requerido.

- **Performance**

El sistema deberá suministrar un desempeño apropiado en relación a los recursos utilizados, con un tiempo de respuesta no mayor a 5 minutos y una base de datos que soporte un máximo de 30GB, con los módulos suficientes requeridos para las PCs de tipo clientes y 2 servidores como mínimo.

- **Soportabilidad**

El sistema tendrá la capacidad de poder ser transferido de un ambiente (organizacional, hardware, software) a otro, y poder tener las siguientes características:

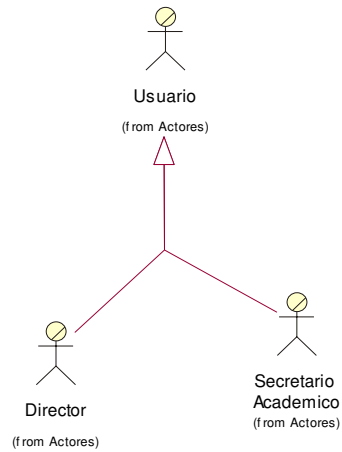
- **Facilidad para adaptarlo:** El sistema trabajara bajo el sistema operativo Windows XP.
- **Facilidad para instalarlo:** En el modulo del cliente se instalara el *.NET Framework*, para la aplicación, en el servidor de aplicaciones *Web*, se tendrá que instalar el *Visual Studio.NET*, y para el servidor de base de datos, se tendrá instalado el SQL Server 2000. Las computadoras necesarias para este sistema tendrán las siguientes características:

- Pentium IV, 1700 MHz, 512 MB RAM, Disco duro 40.

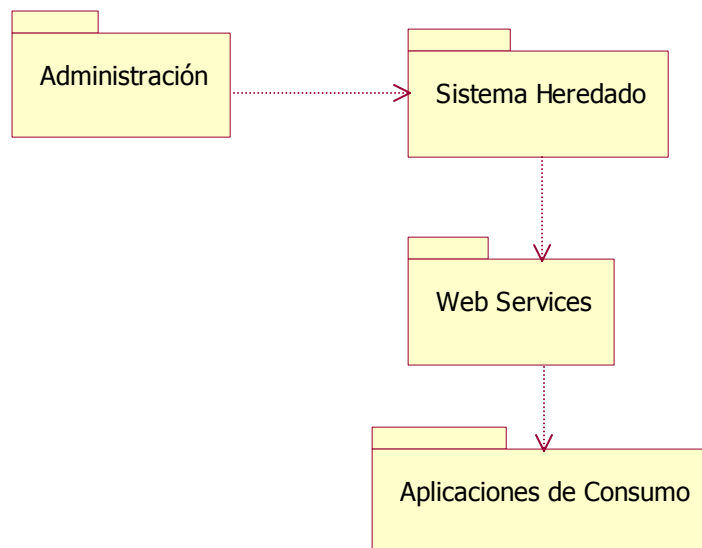
- **Capacidad de reemplazar:** Podrá funcionar bajo versiones mas actualizadas del software requerido.

### 3. Vista de Casos de uso del Sistema

#### a. Relaciones entre actores



#### b. Diagrama de paquetes



## Interfaces del Sistema Heredado

A continuación, se muestran las interfaces que representan al Sistema Heredado, con sus respectivas funcionalidades (Alumnos, Docentes, Cursos, y Carreras, Matricula de Alumnos). Desarrollado en FoxPro para DOS.

### ▪ Menú de Acceso a las funcionalidades del Sistema Heredado

En esta interfaz, se acceden a las funcionalidades del Sistema Heredado. (Ver figura 1).



**Figura 1.** Sistema Heredado: Interfaz de menú principal de la aplicación

## - Interfaz de Apertura del Año Académico

En esta interfaz, se aprecia la funcionalidad del Sistema Heredado con respecto a la apertura del Año Académico. (Ver figura 2).

**ACADEM.EXE**

**E.N.A.M.M** **MODULO ACADEMICO** Mar. 24 de Jun de 2008  
**EDITH** Datos generales del año académico ACA1100

|                                                                                                                                                                                                              |                                                                                                                                                                                                                                        |                                                             |                                                     |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------|-----------------------------------------------------|
| <b>DATOS GENERALES</b><br>Año<br>Descripción<br>Modalidad<br>Responsable                                                                                                                                     | <b>CALIFICACION SEMESTRAL DE CURSOS</b><br>Nota cero<br>No Práctica<br>No exámenes<br>Nt.desap:<br>Nt.exone:                                                                                                                           |                                                             |                                                     |
| <b>DESCRIPCION DE RUBROS</b><br>PS : Promedio semestral académico<br>PP : Promedio de Prácticas<br>PE : Promedio de Exámenes<br>NF : Nota Final del semestre<br>PS : Promedio Semestral<br>EF : Examen Final | <table border="1"> <tr> <td> <b>Promedio semestral</b><br/> <math display="block">PS = \frac{??*PP+??*PE}{???}</math> </td> <td> <b>Nota Final</b><br/> <math display="block">NF = \frac{??*PS+??*EF}{???}</math> </td> </tr> </table> | <b>Promedio semestral</b><br>$PS = \frac{??*PP+??*PE}{???}$ | <b>Nota Final</b><br>$NF = \frac{??*PS+??*EF}{???}$ |
| <b>Promedio semestral</b><br>$PS = \frac{??*PP+??*PE}{???}$                                                                                                                                                  | <b>Nota Final</b><br>$NF = \frac{??*PS+??*EF}{???}$                                                                                                                                                                                    |                                                             |                                                     |

F3 Crear F4 Modificar F5 Eliminar Esc Salir

**Figura 2.** Sistema Heredado: Apertura del Año Académico

## - Interfaz Asignación de Cursos

En esta interfaz, se aprecia la funcionalidad del Sistema Heredado con respecto a la asignación de Cursos. (Ver figuras 3).



**Figura 3.** Sistema Heredado: Interfaz de Asignación de Cursos

#### ▪ Interfaz de Promedio Semestral Académico

En esta interfaz, se aprecia la funcionalidad del Sistema Heredado con respecto al mantenimiento de los docentes. (Ver figura 4).



**Figura 4.** Sistema Heredado: Interfaz de Promedio Semestral Académico

▪ **Interfaz de Impresión de Promedio Semestral Académico**

En esta interfaz, se aprecia la funcionalidad del Sistema Heredado con respecto a la Impresión de Promedio Semestral Académico. (Ver figura 5).

| Nº | Apellidos y Nombres                 | ANALIS.<br>3 | INFORM.<br>3 | ESTADI.<br>4 | NAV |
|----|-------------------------------------|--------------|--------------|--------------|-----|
| 1  | ALAYO PALACIOS, HUMBERTO MANUEL     | 11           | 13           | 10           | 1   |
| 2  | ANGELDONIS BAZALAR, ANGELLO RODOLFO | 13           | 13           | 10           | 1   |
| 3  | AREVALO ACOSTA, MELVIN              | 12           | 12           | 10           | 1   |
| 4  | BALBOA SILVERA, CHRISTIAN DAVID     | 12           | 14           | 14           | 1   |

Estado | Tecla | ↑↓↔PgUpPgDn | Esc | Finalizar

**Figura 5.** Sistema Heredado: Interfaz de Impresión de Promedio Semestral Académico