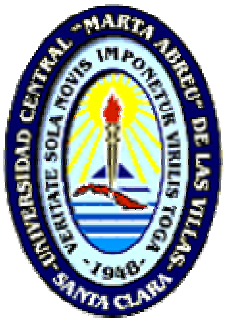
Universidad Central “Marta Abreu” de las Villas

Facultad de Matemática, Física y Computación



Trabajo de Diploma

Arquitectura de Integración

para la

Plataforma de Gestión de ETECSA

Diplomante: Michel González Blázquez

Tutores: Dr. Ricardo Grau Avalos
MSc. Pavel E. Alemán Rodríguez

Santa Clara
2005

DEDICATORIA

A todos los niños que sueñan despiertos. A los que, una vez hombres, continúan soñando. A mi padre.

AGRADECIMIENTOS

A mis amigos de siempre Alcides, Tomy, Eric, Coello, Kiosmy, Disiosky y Portela junto a los cuales he tenido el gusto de aprender computación.

A los trabajadores de ETECSA, que de manera muy entusiasta se han hecho partícipes de este proyecto.

A Jesús por enseñarme la verdadera distancia entre un hombre y sus sueños.

A Pavel, que ha luchado encarnizadamente por mi mejoramiento.

A mi hermana por nunca renunciar a ser ella misma.

A mi novia Ariadni, que con su amor ha logrado llenar cada momento de mi vida.

A mi madre por tantas cosas maravillosas …

RESUMEN

Como parte de la estrategia de la Empresa de Telecomunicaciones de Cuba SA (ETECSA) para orientar su gestión al negocio, está la creación de una gran infraestructura de software. La misma estará compuesta por una gran cantidad de Sistemas de Soporte a la Operación (OSS), que necesitan ser integrados para lograr una gestión extremo a extremo de los servicios. La integración de los sistemas es un proceso complejo que trae implícito un alto riesgo. Dado que existe la necesidad estratégica de avanzar en tiempo récord, la presente tesis propone diseñar una Arquitectura de Integración para la Plataforma de Gestión de ETECSA. Dicha arquitectura tiene como objetivo agilizar la integración de los Sistemas de Soporte a la Operación garantizando la satisfacción de los requerimientos de flexibilidad, fiabilidad y extensibilidad. Para el diseño se utilizaron los patrones de integración como herramienta metodológica, lo que además permitió hacer el análisis y la selección de las tecnologías de implementación. Aparte del diseño presentado, se pone en evidencia la capacidad de los patrones de diseño para resolver problemas complejos.

ABSTRACT

The strategy of the Empresa de Telecomunicaciones de Cuba SA (ETECSA) for a business oriented management, involves the construction of a large software infrastructure. This comprises the integration of Operation Supporting Systems (OSS) for the management side-to-side of services. The integration process is very complex and carries a significant risk, by the order hand; the enterprise needs to do this process a very short term. The current thesis proposes a design of an Integration Architecture for ETECSA's Management Platform to improve the integration process of OSSs at the same time that satisfies the requirements of flexibility, reliability and extensibility. The design is based on Integration Patterns; witch also serves for the evaluation and selection of the implementation technology. Besides the required design, this thesis also focuses on the capacity of patterns in the solution of complex problems.

ÍNDICE

Dedicatoria I
Agradecimientos II
Resumen III
Abstract IV
Índice V
Introducción 1
Capítulo 1. Complejidad 5
1.1 Complejidad de los Sistemas 6
1.2 Nueva Generación de Sistemas de Gestión 8
1.3 Patrones de Diseño 12
1.3.1 Definición de Patrón 13
1.3.2 Forma de un Patrón 13
1.3.3 Catálogos de Patrones 15
1.3.4 Lenguajes de Patrones 15
1.3.5 Diseño basado en Patrones 16
1.4 Conclusiones Parciales 17
Capítulo 2. Patrones de Integración 18
2.1 El Problema de la Integración 18
2.2 Cluster de Patrones de Integración 20
2.3 Patrones de Capas de Integración 23
2.3.1 Capa de Agregación de Entidades 23
2.3.2 Capa de Integración de Procesos 26
2.3.3 Capa de Integración de Portal 31
2.4 Conexiones entre Sistemas 35
2.4.1 Integración de Presentación 36
2.4.2 Integración de Datos 39
2.4.3 Integración Funcional 44
2.4.4 Integración Orientada a Servicios 50
2.5 Topología de Integración 55
2.5.1 Conexión Punto a Punto 56
2.5.2 Broker 56
2.5.3 Bus de Mensajes 58
2.6 Conclusiones Parciales 59
Capítulo 3. Tecnologías de Integración 60
3.1 Microsoft SQL Server Data Transformation Services 61
3.2 Microsoft Message Queuing 63
3.3 COM+ 64
3.4 CORBA 65
3.5 .Net Remoting 67
3.6 ASP.Net Web Services 68
3.7 Microsoft Web Services Enhancements 69
3.8 Indigo 70
3.9 Microsoft Biztalk Server 72
3.10 Conclusiones Parciales 76

Capítulo 4. Arquitectura de Integración 77
4.1 Plataforma de Gestión de ETECSA 78
4.2 Problema de la Fragmentación de la Red 79
4.3 Diseño de la Arquitectura de Integración 81
4.3.1 Automatización de los Procesos de Negocio 82
4.3.2 Orientación de las Aplicaciones a Servicios 84
4.3.3 Integración con los Sistemas de Gestión Legados 86
4.4 Mapa de la Arquitectura de Integración 92
4.5 Análisis de los resultados 93
Conclusiones 95
Recomendaciones 96
Bibliografía 97

INTRODUCCIÓN

El desarrollo acelerado de las tecnologías de comunicación en el último siglo ha cambiado por completo y para siempre el mundo en que vivimos. Las empresas de telecomunicaciones son las que marchan a la cabeza de esta revolución en un negocio que se ha vuelto tan lucrativo como exigente y competitivo.

A pesar de los visibles avances en las tecnologías, la atención personalizada a los clientes es una de las mayores fuentes de ingresos aún no explotada. Es en esta dirección hacia donde han orientado sus mayores y más recientes esfuerzos los operadores de telecomunicaciones, con el objetivo de lograr una diferenciación en el mercado y una mayor captación y control del mismo. Garantizar amplias facilidades de gestión sobre los servicios, es más prioritario que generar nuevas modalidades de estos en un mercado donde los clientes no consumen servicios que no estén respaldados por un amplio espectro de facilidades de soporte. Es por ello que los operadores de telecomunicaciones en el mundo tratan de orientar la gestión de su equipamiento y sus procesos hacia el negocio mediante una mayor oferta de servicios totalmente gestionados, alcanzando una mayor satisfacción del cliente final.

Una gestión de los recursos de la red orientada al negocio requiere de mayores volúmenes de información. Tanta información es inoperante si no se cuenta con las herramientas necesarias para su análisis. Aquí es donde los sistemas de gestión juegan un rol protagónico humanizando el trabajo de los operadores. La interrelación sistema – operador, permite lograr una gestión eficaz de la red que se traduce en tres logros fundamentales para la empresa:

- Aumento de la calidad percibida por el cliente
- Incremento de los beneficios para la empresa
- Optimización de los recursos de la empresa

La Empresa de Telecomunicaciones de Cuba SA (ETECSA) se encuentra enfrascada en lograr una gestión orientada al negocio, con el objetivo de insertarse al más alto nivel entre los operadores de telecomunicaciones de Latinoamérica. Una de las aristas de esta

estrategia es la creación de infraestructuras de software que hagan posible la modernización de sus procesos de negocio. Durante diez años ETECSA ha insertado un conjunto de sistemas conocidos como Sistemas de Soporte a la Operación (OSS del inglés Operation Support System), que han automatizado y centralizado la gestión de la mayoría de su equipamiento de red. El requerimiento del operador en la nueva etapa que comienza, es poder construir sobre estos sistemas una plataforma que le permita manejar de forma integrada y orientada a sus procesos de negocios toda la información de gestión de los servicios que su red de telecomunicaciones soporta.

Desde el punto de vista de la programación, la concepción de una plataforma de gestión posee una gran complejidad. Uno de sus aspectos más difíciles es lograr la necesaria integración de los OSS como parte de un sistema complejo. Si esta integración no es hecha cuidadosamente, toda la plataforma derivaría en un sistema rígido imposible de mantener y mucho menos de modernizar, con lo que se perdería una enorme cantidad de esfuerzo y recursos. Por otro lado, la empresa necesita contar cuanto antes con dicha plataforma para poder asumir los retos que se ha propuesto. Esta disyuntiva ha llevado a la necesidad de efectuar un estudio de las principales tendencias para la solución de dicho problema y de contar con una arquitectura que brinde soporte a la integración de los OSS dentro de la plataforma de gestión.

El diseño de una arquitectura de integración depende fundamentalmente de que se logre dominar la complejidad intrínseca de los problemas de integración. Experiencias preliminares en ETECSA [Rodríguez03], muestran la capacidad de los patrones de diseño para abordar problemas complejos. Por tanto, se decidió basar el diseño de la arquitectura en un conjunto de Patrones de Integración.

El diseño deberá tomar en cuenta las tendencias actuales de los sistemas de gestión de las telecomunicaciones y efectuar un análisis de las metodologías y tecnologías más modernas desde el punto de vista de la integración.

De acuerdo a lo anterior, las principales preguntas de investigación son las siguientes:

1. ¿Tienen los patrones de integración existentes las capacidades suficientes para el diseño de una arquitectura como la deseada?
2. ¿Existen suficientes tecnologías y herramientas profesionales para implementar rápidamente una arquitectura de integración como la deseada?
3. ¿Es posible combinar los patrones de integración y las herramientas para el diseño de la arquitectura de integración para la Plataforma de Gestión de ETECSA?

Después de la revisión de la literatura, las hipótesis o presuntas respuestas a las dos primeras preguntas de investigación resultaron positivas y se conjeturó entonces la respuesta también positiva para la tercera. La principal hipótesis se concentra en la posibilidad de combinar los patrones y las tecnologías en el diseño e implementación de la arquitectura. Consecuentemente, el objetivo general de la tesis se formula así:

Diseñar una Arquitectura de Integración para la Plataforma de Gestión de ETECSA mediante el uso de patrones de integración y con las tecnologías disponibles, con el fin de agilizar la integración de los Sistemas de Soporte a la Operación en dicha empresa garantizando la satisfacción de los requerimientos de flexibilidad, fiabilidad y extensibilidad.

El cumplimiento final de este objetivo supuso la realización de las tareas siguientes:

1. Revisión del estado del arte de la gestión de las telecomunicaciones y los Sistemas de Soporte a la Operación (OSS).
2. Determinación de las ventajas del diseño con patrones para la solución de problemas complejos.
3. Revisión del estado del arte de los patrones de integración.
4. Evaluación de las tecnologías disponibles para la implementación de la Arquitectura de Integración.
5. Elaboración del diseño de la Arquitectura de Integración y de la propuesta de tecnologías necesarias para su implementación

La tesis quedó finalmente estructurada en 4 capítulos. Las tareas 1 y 2 son básicas y se exponen en el Capítulo 1. Los resultados de la tarea 3, relativa ya específicamente a la

descripción de los patrones de integración, se expone en el Capítulo 2. El capítulo 3 se dedica a la evaluación de las tecnologías (tarea 4) y finalmente el diseño de la arquitectura se expone en el capítulo 4, verificando así la hipótesis principal.

CAPÍTULO 1. COMPLEJIDAD

La necesidad de orientar la gestión de la red al negocio, ha provocado cambios en las estructuras informativas de las empresas de telecomunicaciones. Las informaciones referentes al mercado, los clientes, los servicios y la subcontratación de terceros, se han sumado al tradicional manejo de los recursos de la red. La gestión actual ha incorporado nuevas dimensiones y se ha tornado más compleja.

Los Sistemas de Soporte a la Operación juegan un papel fundamental dentro de las empresas a la hora de alcanzar una gestión a la altura de las exigencias del mercado. Las empresas de telecomunicaciones demandan constantemente de Tecnologías de la Información con el objetivo de ganar en competitividad.

La aplicación de Tecnologías de la Información requiere del entendimiento de los procesos de gestión. Para los especialistas, no es una tarea fácil modelar el complejo flujo de información y el enorme número de operaciones que intervienen en ellos. Es preciso contar con una metodología que permita atacar la complejidad del problema, si es que se quiere tener éxito.

Una vez que se ha comprendido el problema, el reto consiste en diseñar e implementar las aplicaciones. La creciente complejidad de la gestión da al traste con la idea de crear grandes sistemas que den soporte a todas las operaciones. Necesariamente estos fueron sustituidos por sistemas más pequeños especializados en cada una de las capas funcionales. Con posterioridad, la idea de los pequeños sistemas de gestión aislados evolucionó hacia el concepto de Plataforma de Gestión, en la cual, varios sistemas cooperan logrando el funcionamiento de la empresa.

Antes de comenzar a diseñar con lujo de detalles una infraestructura de grandes proporciones como lo es una Plataforma de Gestión, es importante elaborar un diseño a grandes rasgos de lo que se pretende. Esta etapa temprana del diseño se diferencia fundamentalmente de las otras, por la inexperiencia de los programadores y el corto tiempo de que se dispone. A pesar de ello, el resultado de este proceso tiene una gran repercusión sobre las decisiones que toman los directivos para trazar la estrategia hacia el objetivo.

La etapa temprana del diseño se caracteriza por el alto grado de abstracción con que aborda el problema y la solución. En la solución obtenida, los programadores deben reflejar los aspectos fundamentales del problema. Esta solución, debe acercarse lo más posible a la solución real que se obtenga posteriormente de un diseño detallado.

El objeto del presente capítulo es la determinación de las técnicas apropiadas para abordar el diseño de la Arquitectura de Integración en su etapa temprana. En consecuencia se hace un análisis de las características de los sistemas complejos y algunos de los métodos para su modelación, tanto de manera general, como en el marco de la gestión de las telecomunicaciones. Como resultado, se seleccionan las metodologías a utilizar en los subsiguientes desarrollos de la tesis.

1.1 Complejidad de los Sistemas

A medida que las infraestructuras de software evolucionan acumulan complejidad. La complejidad está intrínsecamente ligada al hecho de que se quiere que el software se ajuste cada vez más a los problemas reales, que son inherentemente complejos dada la cantidad de elementos que participan y la naturaleza de sus interacciones. Gran parte del esfuerzo creativo del hombre en la programación se concentra en dominar la complejidad. Las metodologías utilizadas en esta labor han ido evolucionando hacia formas que se ajusten a la manera de pensar de los seres humanos.

Se reconoce que, en el desarrollo de software, un gran salto en el sentido metodológico se produjo con el surgimiento de la Programación Orientada a Objetos. Es en está forma de análisis y diseño donde se encuentran las bases teóricas y metodológicas más sólidas para el dominio de la complejidad. Según Graddy Booch [Booch98] un sistema complejo se caracteriza por cinco atributos:

1. Frecuentemente, la complejidad toma la forma de una jerarquía en la cual un sistema complejo se compone de subsistemas relacionados, que tienen a su vez sus propios subsistemas y así sucesivamente hasta que se alcanza algún nivel ínfimo de componentes elementales.

El hecho de que muchos sistemas complejos tengan una estructura jerárquica y descomponible es un factor decisivo, pues permite construir sistemas complejos a través la de elaboración de unos cuantos componentes fundamentales y su posterior ensamblaje en creciente grado de complejidad.

2. La elección de qué componentes de un sistema son primitivos es relativamente arbitraria y queda a elección del observador (…) Lo que es primitivo para un observador puede estar a un nivel de abstracción mucho más alto para otro.

La elección de las abstracciones a partir de las cuales se comienzan a estructurar los sistemas, es fundamental. Un buen proceso de abstracción reduce significativamente la complejidad de los sistemas al descartar los detalles irrelevantes y por otra parte conservar aquellos que son definitorios.

3. Los enlaces internos en los componentes suelen ser más fuertes que los enlaces entre los componentes. Este hecho tiene el efecto de separar la dinámica de alta frecuencia de los componentes, que involucra a la estructura interna de los mismos, de la dinámica de baja frecuencia, que involucra la interacción entre los componentes.

La diferencia entre interacciones intracomponentes e intercomponentes proporciona una separación clara de intereses entre las diferentes partes de un sistema, posibilitando el estudio y desarrollo de cada parte de forma relativamente aislada. A su vez, tiene implicaciones en el plano tecnológico, pues en muchos casos define la forma que tendrá la arquitectura final una vez elaborada.

4. Los sistemas jerárquicos están compuestos usualmente de solo unas pocas clases diferentes de subsistemas en vanas combinaciones y disposiciones.

En otras palabras, todos los sistemas complejos poseen patrones comunes. Entre dos sistemas jerárquicos, será mejor aquel cuyos subsistemas sean más simples y aparezcan con más frecuencia en diferentes combinaciones o disposiciones. Esto es una especie de principio de “parsimonia” en la determinación de patrones. La existencia de estos patrones

es lo que posibilita la reutilización de los diseños y componentes. Este hecho, se verá con más claridad en los epígrafes subsiguientes.

5. Se encontrará invariablemente que un sistema complejo que funciona ha evolucionado de un sistema simple que funciona. Un sistema complejo diseñado desde cero nunca funciona y no puede parchearse para conseguir que lo haga. Hay que volver a empezar, partiendo de un sistema simple que funcione.

Esta última característica es la máxima que siguen las metodologías modernas de desarrollo y mantenimiento del software. El cómo lograr primero pequeños sistemas funcionales y posteriormente evolucionar hacia niveles crecientes de funcionalidad, es la clave para el desarrollo exitoso de sistemas de mediana y gran complejidad.

1.2 Nueva Generación de Sistemas de Gestión

A medida que la separación entre el mundo de las comunicaciones de datos y el mundo de las telecomunicaciones desaparece rápidamente, la complejidad y el tamaño de las redes que soportan la naciente industria de las comunicaciones crece. La capacidad de los sistemas existentes para manejar las redes de información y comunicación está siendo rápidamente agotada por el crecimiento y la diversificación de las redes, la complejización de las aplicaciones, el aumento del número de usuarios y la sofisticación de los servicios. Para gestionar la nueva generación de redes y servicios se hace necesario una nueva generación de sistemas informáticos. [TMF04]

Uno de los trabajos más serios en la solución de los problemas actuales de los sistemas de gestión lo lleva a cabo el TeleManagement Forum (TMF), una organización formada por más de 400 compañías y donde se encuentran una buena parte de los líderes mundiales de las telecomunicaciones y las tecnologías de la información. Su proyecto NGOSS (New Generation of Operating Systems and Softwares) tiene como objetivo la concepción de una infraestructura que permita a los programadores concebir una nueva generación de sistemas de gestión orientados al negocio.

Con NGOSS se pretende construir sistemas de gestión que logren satisfacer las necesidades siguientes [TMF05]:

1. Mayor automatización de los procesos de la empresa.
2. Rápida implementación de nuevos servicios.
3. Comercio B2B más fácil y flexible.
4. Menores costos de adquisición.
5. Menores costos ante los cambios.
6. Mayor cantidad de servicios.
7. Mejor calidad de los servicios.
8. Mejor flujo de los procesos de la empresa.
9. Mejor control de los clientes.
10. Una provisión más flexible de los servicios.

Las directivas de trabajo de NGOSS pueden clasificarse en cuatro grandes áreas como se muestra en la Figura 1.1.

1. El análisis de los procesos de negocio.
2. El análisis y diseño de sistemas.
3. El análisis de las soluciones.
4. La conformidad con los estándares, las pruebas y la implantación.

Cada una de estas áreas constituye una etapa en el ciclo de desarrollo de nuevos sistemas. En cada área se trabaja en el marco de un proyecto, los cuales son:

1. Mapa Operacional de las Telecomunicaciones (eTOM del inglés Enhanced Telecom Operations Map).
2. Modelo de Información Compartido (SID del inglés Shared Information Data Model).
3. Arquitectura Neutral a la Tecnología (TNA del inglés Technology Neutral Arquitecture, también llamado GNOSS Arquitecture).
4. Pruebas de Conformidad.

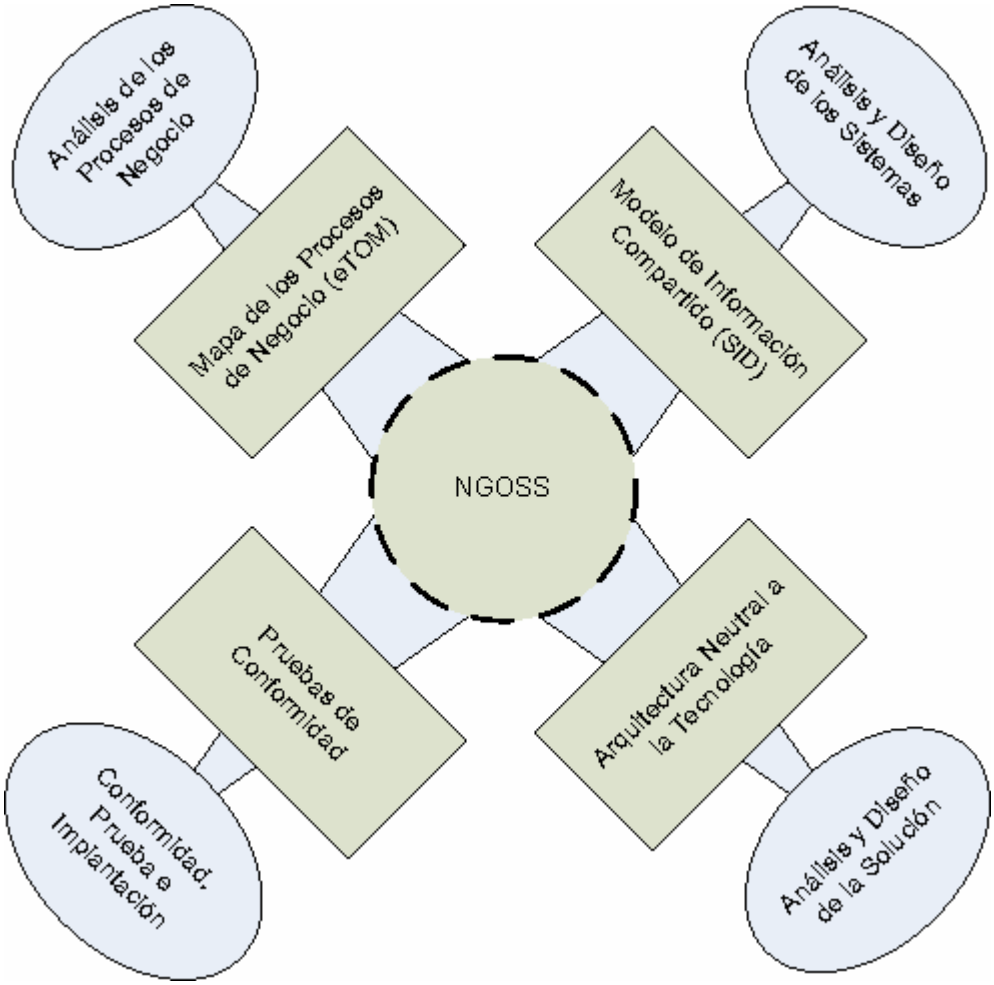


Figura 1.1 Proyecto NGOSS

El proyecto eTOM (Enhanced Telecom Operations Map) es el sucesor de TOM (Telecom Operations Map). En su primera versión TOM presenta un mapa funcional de las empresas de telecomunicaciones, en donde están representados todos los procesos funcionales a diferentes niveles de abstracción. Usando este mapa como punto de partida, los programadores pueden explorar la lógica del negocio de las telecomunicaciones. Por lo tanto es una pieza fundamental para la comprensión y concepción de sistemas de gestión.

El enfoque de TOM tiene una gran dificultad, al solo modelar las funciones de cada una de las capas de la empresa, no tiene en cuenta el flujo de las informaciones entre dichas capas. Como resultado, cada capa funcional en TOM es capaz de manejar sus informaciones pero no existe correlación entre las informaciones de diferentes capas.

Por otra parte las estrategias de marketing más modernas, las cuales incluyen la personalización de los servicios de atención al cliente, arrojan la necesidad de dar un seguimiento al flujo de las informaciones desde la capa de atención a los clientes hasta la capa de operación de la red. Esta necesidad no es satisfecha por TOM debido a la imposibilidad de correlacionar las informaciones entre las diferentes capas funcionales.

Con eTOM se hace una nueva división funcional de la empresa agregándole el flujo de las informaciones entre las capas. Dicho flujo recibe el nombre de **proceso de negocio**. Un proceso de negocio en eTOM describe las interacciones entre cada una las unidades de la empresa a diferentes niveles de abstracción. Con ello se completa el mapa de las empresas de telecomunicaciones abriendo el camino para una nueva área de trabajo, la automatización de los procesos de negocio.

Otro de los proyectos de gran repercusión dentro de NGOSS es SID (Shared Information/Data Model). Mediante el cual se intenta dotar tanto a operadores de telecomunicaciones como programadores de un lenguaje conceptual común. El proyecto hace un estudio de los conceptos presentes en la gestión de las telecomunicaciones y propone una expresión de los mismos en las entidades de datos de cada uno de los sistemas de gestión. Por ello SID está considerado como el puente que conecta tres grandes áreas de trabajo de NGOSS: el negocio, los sistemas y las soluciones.

Mediante eTOM se defiende la estandarización de los procesos de gestión, con SID se trabaja en la estandarización de las informaciones y sistemas de gestión. La estandarización es una de las claves de NGOSS para producir sistemas altamente compatibles que permitan una fácil integración inter-empresarial. Esto estimula los negocios y la subcontratación de servicios entre las empresas de telecomunicaciones.

El tercer proyecto está dedicado a establecer las pautas desde el punto de vista técnico para la implementación de los sistemas de gestión. Este proyecto propone una arquitectura que es lo suficientemente general como para ser implementada usando una amplia gama de tecnologías. Mediante la neutralidad se trata de que cada empresa continúe utilizando la línea tecnológica a la que está habituada. A su vez esta arquitectura debe garantizar la interoperabilidad de las aplicaciones con independencia de su tecnología.

Cualquier actividad de estandarización como la que propone NGOSS requiere de herramientas que ayuden a medir el nivel de conformidad de la solución con las especificaciones. El cuarto proyecto de NGOSS está orientado a la creación de pruebas y herramientas para ello. Las pruebas de conformidad le son útiles tanto a los productores de sistemas de integración como a los posibles compradores, ya que constituyen una manera científica de validar los sistemas.

Un análisis de las herramientas de NGOSS permite concluir que las mismas atacan la complejidad de la gestión utilizando los mismos mecanismos expuestos por Graddy Booch [Booch98] y analizados anteriormente. La herramienta eTOM divide el sistema de complejo en subsistemas más simples y explica las relaciones entre ellos a través de los procesos. Por su parte, SID auxilia a los programadores en la búsqueda de las abstracciones primitivas y su organización jerárquica. Mediante el uso de dichas herramientas se ahorra una parte significativa del tiempo dedicado al diseño a la vez que se gana en fiabilidad.

Al asumir NGOSS como guía para la concepción de sus plataformas de gestión, las empresas de telecomunicaciones se nutren con una basta experiencia, que no es más que el resultado coordinado de muchas empresas en esta área. Desde el punto de vista de la empresa como desarrolladora de su propia infraestructura, la metodología, los modelos, las especificaciones y los patrones expuestos por NGOSS contribuye a crear sistemas más robustos y enfocados a los principios más modernos de la gestión. Desde el punto de vista de la empresa como comprador de un sistema, las pruebas de conformidad le brindan una herramienta para determinar si el sistema propuesto se ajusta en su arquitectura de software.

1.3 Patrones de Diseño

En la literatura se destaca la existencia de patrones en el software [Booch98]. Los patrones, al igual que la Programación Orientada a Objetos, son un reflejo de los mecanismos del pensamiento, por lo que no es difícil encontrarlos en los procesos de diseño de casi todas las ramas de la ingeniería. De hecho, no ha de sorprender que sus primeros enfoques teóricos surgieran en una de las más antiguas, la arquitectura.

El interés por los patrones en el software surgió entre los líderes del diseño de software en la industria. Emergieron directamente del diseño orientado a objetos, hecho natural, pues la

Programación Orientada a Objetos había sido el foco de atención del diseño de software desde mediados de los 80. Aunque el vínculo entre los Patrones y la Programación Orientada a Objetos persiste hoy en día, no existe nada intrínseco en esta relación, ya que los patrones existen en muchos otros contextos.

La utilización de patrones, como se analizará más adelante, hace más factibles los procesos de diseño de software al tornarlos más sencillos, rápidos y seguros. Este hecho que argumenta el por qué de su posterior popularización hasta constituir, hoy en día, uno de los principales focos de atención de los productores de software industrial.

1.3.1 Definición de Patrón

"Cada patrón es una regla que expresa la relación entre tres partes: un cierto contexto, un problema y una solución." [Alexander79]

O sea, cada patrón hace un análisis de un problema más o menos abstracto dentro de un contexto determinado y propone una solución que resuelve en parte o completamente dicho problema.

Un patrón también puede definirse como:

"Un fragmento de literatura que describe un problema y una solución general de ese problema en un contexto determinado." [Coplien00]

En la segunda definición, aunque no es lo suficientemente general, sí se destaca el hecho que los patrones se escriben siguiendo un determinado esquema literario llamado **forma**. La forma literaria de los patrones ayuda a los programadores a transmitir las ideas a sus clientes u otros programadores, resolviendo así uno de los principales problemas en el desarrollo de software, la comunicación.

1.3.2 Forma de un Patrón

Los patrones se pueden ver como una forma literaria y como consecuencia existen toda una diversidad de estilos. Los estilos o formas más populares de los patrones son: la forma Alejandrína, la forma de GOF (Gang of Four), la forma de Portlan y la forma de Coplien [Coplien00].

Todas estás formas giran en torno a un subconjunto de secciones:

- *Nombre:* Nombre del patrón.
- *Objetivo:* Resume, en una frase u oración, lo que el patrón hace.
- *Problema:* Describe un problema general a ser resuelto.
- *Contexto:* Describe explícitamente el contexto en que se desarrolla dicho problema. Hace un recuento de los patrones que han sido aplicados antes que el patrón en cuestión sea considerado. Especifica el tamaño, alcance, mercado, lenguaje de programación y otras características, que de cambiar, pueden invalidar el patrón.
- *Fuerzas:* Analiza detalladamente el problema al dividirlo en partes (fuerzas) y explicar sus interacciones. Esta sección es sumamente importante pues establece los requisitos que la solución deberá cumplir a través del correcto "balance de las fuerzas" que intervienen en el problema.
- *Solución:* Provee una solución lo suficientemente detallada tal que indique a los diseñadores qué hacer. Dicha solución no es única y casi siempre es una idea general pues queda por parte del diseñador ajustarla al problema específico.
- *Diagrama:* Provee un diagrama de la solución que sintetiza el patrón para una rápida comprensión.
- *Contexto resultante:* Describe el contexto resultante respondiendo, entre otras, a las preguntas siguientes: ¿Qué fuerzas fueron resueltas? ¿Qué nuevos problemas se derivan de la aplicación del patrón? ¿Qué otros patrones pueden aplicarse a continuación?

Las secciones en los patrones tienen la función de organizar y formalizar la información facilitando la búsqueda y el manejo de la misma. El subconjunto de secciones básicas expuesto anteriormente pueden ser enriquecido con implementaciones, referencias a tecnologías, ejemplos, etc.

En los patrones que se expondrán en lo siguiente, se ha limitado la escritura a un subconjunto básico de secciones. Para una completa y detallada descripción de los mismos se puede recurrir a los catálogos de patrones y la literatura especializada [Townbridge03], [Hohpe03], [Townbridge04], [Patternshare05].

1.3.3 Catálogos de Patrones

Los catálogos de patrones contienen un conjunto de patrones que casi siempre son afines a un problema o contexto general. Estos catálogos constituyen una guía para el desarrollo de aplicaciones, pues cada patrón contiene un mecanismo simple pero probado para resolver un pequeño problema [Trowbridge03].

Por ejemplo, un desarrollador tiene un problema que intenta solucionar. Primero consulta el catálogo en busca de un patrón que le sugirieron, o simplemente analiza un resumen de los objetivos de cada patrón en busca de alguno que se ajuste a grandes rasgos a su caso. Después que encuentra un patrón posiblemente útil, procede a la lectura del problema y el contexto para determinar si realmente se ajusta a su problema y contexto específico. Si el patrón se ajusta, entonces el desarrollador avanza a la lectura de las fuerzas. En esta sección aprende más sobre su problema y lo comprende mejor, nota que existen análisis que todavía no ha hecho; en pocas palabras, se nutre con la experiencia acumulada sobre el problema. En este punto, el desarrollador puede tomar la solución propuesta u optar por una propia. Finalmente toma nota del contexto resultante, analiza hasta que punto solucionó su problema y determina los próximos pasos que pudiera dar.

El ejemplo anterior ilustra un algoritmo, a partir del cual alguien con un conocimiento más o menos vago sobre un problema que intenta resolver puede auxiliarse de un catálogo de patrones en busca de una solución. Sin embargo, esta forma “secuencial” de buscar no es óptima, existen otras formas más avanzadas de organizar y buscar patrones como los clusters de patrones, los cuales se analizarán posteriormente.

1.3.4 Lenguajes de Patrones

“Un lenguaje de patrones es una colección de patrones que se estructuran para generar un sistema. Un patrón aislado resuelve un problema aislado; un lenguaje de patrones construye un sistema. Es a través de los lenguajes de patrones que los patrones alcanzan todo su poder.” [Coplien00]

Los lenguajes de patrones defienden la idea de exponer los sistemas como una combinación de patrones. Este enfoque tiene grandes implicaciones en la estructuración de un sistema altamente compresible y documentado ya que se apoya en patrones bien establecidos y

conocidos por los desarrolladores. Y lo mejor de todo, a muy bajo costo pues gran parte de la elaboración de la documentación son simples referencias a dichos patrones.

Haciendo una analogía con el lenguaje natural en el que las palabras se estructuran hasta formar oraciones con un significado; los patrones se estructuran hasta formar arquitecturas de propósito específico [Coplien00]. Esta afirmación ha conducido a una confusión con los lenguajes de programación, sin embargo los lenguajes de patrones trabajan a un nivel mucho más alto de abstracción, el plano de las ideas de diseño.

Los lenguajes de patrones establecen relaciones entre los patrones para resolver un conjunto de problemas en torno a un escenario. De dichas relaciones se puede destacar la siguiente:

Refinamiento: Es la relación que se establece entre un patrón que resuelve un problema general y un patrón que resuelve un caso específico del problema inicial. En otras palabras, la relación que se establece entre el patrón "abstracto" y en patrón "concreto" se llama **refinamiento**[1]. Después que se conoce un patrón abstracto solo basta con comprender las diferencias incrementales introducidas en sus sucesivos refinamientos [Trowbridge03].

Las relaciones entre los patrones en un lenguaje de patrones conforman un grafo direccionado y acíclico que permite la navegación desde un patrón hacia otros. Sin embargo, algunos patrones están más "cerca" de otros y esto lleva a la idea de organizar los patrones "más cercanos" en **clusters de patrones**. Los cluster de patrones agrupan patrones por áreas de acción que resultan de dividir la construcción de sistemas usando diversos criterios como: nivel de abstracción, problemas generales, roles dentro del desarrollo, etc.

1.3.5 Diseño basado en Patrones

A la utilización extensiva de los lenguajes de patrones durante el proceso de diseño se le conoce como **diseño basado en patrones**. Este método se apoyan en el análisis del problema y las soluciones propuestas en los patrones. El diseño basado en patrones posee un gran número de ventajas entre las cuales se destacan las siguientes:

[1] Durante el texto también se utiliza el término refinamiento para referirse al patrón concreto de una relación de refinamiento entre dos patrones.

- Los patrones proveen un estudio condensado de los principales aspectos del problema, resultado de la experiencia acumulada de los expertos en el tema.
- Los patrones documentan mecanismos simples y funcionales que sirven como punto de partida para la construcción de estructuras más grandes.
- Los patrones proveen un vocabulario y una taxonomía común a desarrolladores y arquitectos de software contribuyendo en gran medida a la comunicación.
- El uso de patrones incrementan la reutilización de las arquitecturas, los diseños y las implementaciones.
- Los lenguajes de patrones permiten aumentar el poder solucionador de los patrones.
- Los lenguajes de patrones permiten que los diseños sean descritos brevemente como una combinación de patrones. Esto conduce a diseños muy comprensibles y documentados a muy bajo costo, una consecuencia fundamental en la obtención de sistemas mantenibles e independientes del grupo inicial de desarrolladores.
- Los lenguajes de patrones explican la naturaleza de las interacciones entre los patrones indicando que patrones combinar, o que patrones más concretos revisar.
- Los lenguajes de patrones conllevan a la formación de clusters que reducen la longitud de la búsqueda a un área de acción determinada.

1.4 Conclusiones Parciales

- El proyecto NGOSS ofrece una metodología y un conjunto herramientas para atacar la complejidad del problema de la gestión de las telecomunicaciones.
- Al asumir NGOSS como guía para la concepción de sus Plataformas de Gestión, las empresas de telecomunicaciones se apoyan en el trabajo realizado. La experiencia acumulada y los estándares propuestos por los líderes en el tema conllevan a una mayor fiabilidad en los procesos de diseño y desarrollo, con un consecuente ahorro de tiempo y recursos.
- La utilización del diseño basado en patrones es sumamente ventajosa para la solución de problemas complejos, fundamentalmente durante la etapa temprana del diseño.

CAPÍTULO 2. PATRONES DE INTEGRACIÓN

El perfeccionamiento de las redes y la realidad de Internet están cambiando la concepción sobre las aplicaciones actuales de manera radical. Hoy en día las aplicaciones no existen aisladas, sino que están conectadas a fuentes de datos y servicios de otras aplicaciones. Las redes internacionales de bancos, el comercio electrónico y las tiendas virtuales son ejemplos de la potencialidad del software en un futuro altamente interconectado, donde una aplicación sería solo una pieza en la inmensa red en la que las aplicaciones colaborarían para resolver problemas.

El presente capítulo describe el problema de la integración de aplicaciones de manera general. Posteriormente, procede a presentar sistemáticamente los principales patrones que conforman el cluster de Patrones de Integración propuesto por Trowbridge [Trowbridge04]. A través de dichos patrones se resume el estudio de las problemáticas de la integración y se propone un conjunto de soluciones, que conforman la base para el diseño de la Arquitectura de Integración.

2.1 El Problema de la Integración

La forma en que las aplicaciones se integran para resolver un problema dentro de un contexto determinado, recibe el nombre de **solución de integración**. Paralelamente, todo lo que proporciona soporte para el modelo de la solución de integración pero que no forma parte del mismo, constituye la **arquitectura de integración**. Esta arquitectura, como se verá más adelante, es de singular relevancia pues establece las leyes o reglas que siguen las aplicaciones para integrarse.

La mayoría de las empresas construyen soluciones de integración muy complejas de manera bastante predecible: las pequeñas unidades dentro de la empresa, como los departamentos o las divisiones, avizoran la capacidad de aplicar Tecnologías de la Información (IT del inglés *Information Technologies*) a un problema. Estas unidades financian proyectos con vistas a crear aplicaciones que exploten dicha capacidad; sin embargo, prestan poco o ningún interés en la arquitectura técnica resultante siempre y cuando se cumpla con los requerimientos de negocio establecidos [Trowbridge04].

Apurarse demasiado en dar respuesta a las necesidades sin detenerse en la forma que éstas soluciones encajan dentro de una Arquitectura de Integración**,** deviene en un cúmulo de aplicaciones arquitecturalmente incompatibles y un alto costo a la hora de crear o modificar las mismas. Lo que es mucho peor, la empresa se pone en desventaja con relación a otros competidores, que al poseer arquitecturas bien construidas, pueden dar soluciones mucho más rápidas a las cambiantes necesidades del mercado.

Por otro lado, la indulgencia ante la tendencia natural de los ingenieros a estudiar el problema profundamente antes de actuar, puede provocar soluciones demasiado lentas. Si el período de desarrollo no es cuidadosamente controlado, se corre el riesgo de perder la oportunidad de obtener una ganancia y con ello, todos los esfuerzos acometidos hasta el momento serían vanos.

Por tanto, encontrar un balance adecuado entre las necesidades de la tecnología y las del negocio es bastante delicado, por lo que es aconsejable que se haga uso, siempre y cuando que sea posible, de estudios y mecanismos validados con anterioridad.

Las arquitecturas de integración tienen que enfrentarse a varios retos fundamentales [Hohpe03]:

- *Las redes no son confiables:* Las arquitecturas de integración tienen que transportar datos de una computadora hacia otra a través de redes. Comparado con un proceso que corre en una sola computadora, la **computación distribuida** debe estar preparada para lidiar con un conjunto mucho más extenso de problemas. Por ejemplo, al integrar dos sistemas que están separados geográficamente por continentes y cuyos datos tienen que viajar a través de líneas telefónicas, redes de área local, redes públicas y enlaces satelitales, debe tenerse en cuenta los retardos e interrupciones que introducen cada uno de estos elementos.
- *Las redes son lentas:* El envío de datos a través de las redes es varias veces más lento que hacer una llamada a un procedimiento local. Diseñar soluciones distribuidas de la misma forma en que se diseña una aplicación aislada puede tener implicaciones de rendimiento desastrosas.

- *No existen dos aplicaciones tecnológicamente iguales:* Las arquitecturas de integración tienen que posibilitar el flujo de información entre sistemas que usan diferentes lenguajes de programación, sistemas operativos y formatos de datos. Una arquitectura de integración tiene que ser capaz de conectar tecnologías diferentes, que en muchos casos no están siquiera diseñadas para ello.
- *El cambio es inevitable:* Las aplicaciones cambian con el tiempo y la arquitectura de integración debe mantener estabilidad de las aplicaciones ante estos cambios. Si el cambio de un sistema provoca que todos los sistemas restantes tengan que ser cambiados, las soluciones de integración pueden caer fácilmente en una avalancha inmantenible de cambios. Por tanto, una buena arquitectura de integración debe reducir el impacto provocado por los cambios minimizando las dependencias entre los sistemas mediante un **acople débil** entre las aplicaciones.

Por otra parte, Trowbridge [Trowbridge04] agrega que uno de los principales obstáculos a sobrepasar cuando se integran aplicaciones es el fenómeno de la **disonancia semántica**. Por disonancia semántica se entiende la situación donde datos de diferentes aplicaciones que aparentemente son los mismos no necesariamente tienen el mismo significado. Las disonancias semánticas son en muchas ocasiones difíciles de detectar y reconciliar. Algunos tipos de soluciones de integración requieren una resolución completa de las disonancias semánticas (lo que en ocasiones puede ser imposible), mientras que otros se acomodan mejor a las ambigüedades.

2.2 Cluster de Patrones de Integración

Existen múltiples autores que han abordado el tema de la integración y de diversas maneras han propuesto un conjunto de patrones de diseño. Sin embargo, pocas fuentes han realizado un trabajo tan serio en la clasificación y organización de los patrones de integración en como el de Townbrigde [Townbridge04]. El cluster de patrones propuesto por este autor se muestra en la Figura 2.1.

El cluster propuesto por Townbridge aborda la problemática de la integración separándola en tres áreas (subclusters):

1. *Capas de Integración:* Sigue el principio de la separación de las soluciones de integración en una capa independiente de las aplicaciones.
2. *Conexiones entre Sistemas:* Estudio de las formas de interconexión entre los sistemas.
3. *Topología de Integración:* La estructura con que se organizan los elementos dentro de las soluciones de integración para formar un todo coherente.

Tomando el cluster como base para el análisis sistémico de los patrones, la presente tesis hace un estudio condensado de los patrones que se utilizaron en el diseño de la arquitectura así como de otros patrones y conceptos que se consideran importantes para el entendimiento del escenario general o que pudieran tener una aplicación en versiones futuras.

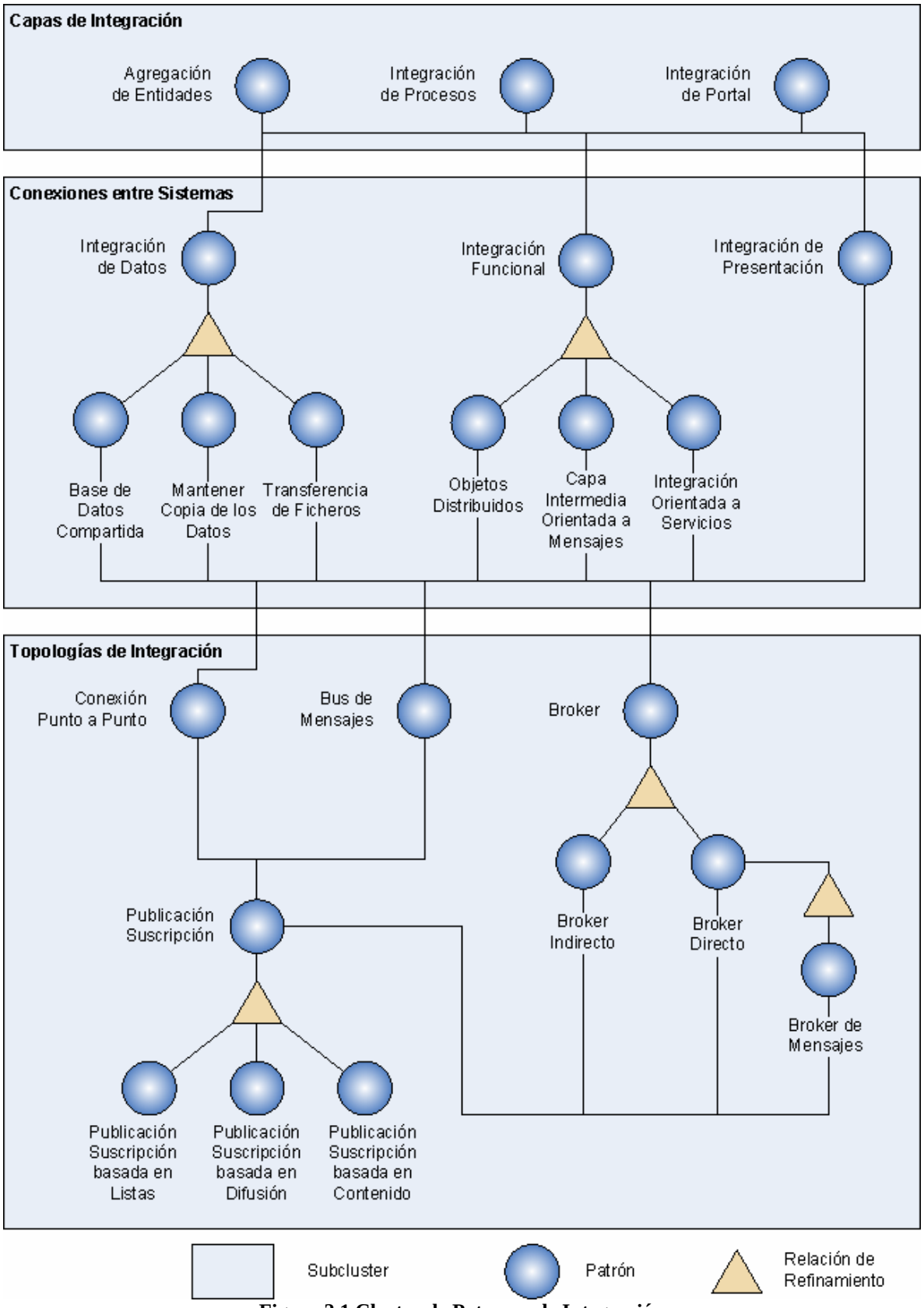


Figura 2.1 Cluster de Patrones de Integración

2.3 Patrones de Capas de Integración

La concepción de una arquitectura de integración parte de la separación de las problemáticas de la integración del modelo de las aplicaciones. Esta separación a menudo toma la forma de una capa[2] que brinda servicios de integración a las aplicaciones.

En la literatura [Townbridge04] se mencionan tres tipos de capas de integración: la capa de **agregación de entidades**, la capa de **integración en portal** y la capa de **integración de procesos**. Cada una de ellas está basada en un patrón de diseño diferente que establece el qué, el cómo y el cuándo de su aplicación.

2.3.1 Capa de Agregación de Entidades

Nombre

Agregación de Entidades

Contexto

Los datos a nivel de empresa están distribuidos a lo largo de múltiples repositorios de forma inconsistente. Las aplicaciones existentes necesitan una representación única y consistente de las entidades de datos. El movimiento de datos a través de los repositorios puede no ser una opción viable.

Problema

¿Cómo pueden ser mantenidos por las aplicaciones, los datos distribuidos redundantemente a través de múltiples repositorios?

Fuerzas

- Pueden existir múltiples sistemas que actúen sobre las mismas entidades de datos. En cada uno, el modelo de negocio establece la forma en que se definen y se actualizan las entidades. Por ejemplo, la entidad Empleado es comúnmente definida por el Sistema de Manejo de Recursos Humanos, por el Sistema de Nómina, así como por otros sistemas. Sin embargo, al construir un sistema de servicios para los empleados se necesita tener una vista completa de lo que constituye un empleado y no el conjunto de partes por separado.

[2] Se refiere a la construcción de aplicaciones o arquitecturas estratificadas, en las cuales las capas inferiores brindan servicios a las capas superiores.

- A menudo, existe disonancia semántica entre los datos de los diferentes repositorios. Lo que aparentemente es el mismo dato puede no representar la misma información. Por ejemplo el dato NúmeroDeDíasRestantes para un proyecto puede incluir los días festivos en un sistema; sin embargo en otro, solamente incluye los días laborables.
- Aún cuando los datos son semánticamente consistentes, la información que representan puede variar paralelamente en las múltiples instancias del dato. En estos casos es difícil determinar qué instancia contiene la información correcta.
- Pueden ser introducidos datos no válidos por otros puntos de entrada a los repositorios, debido a que no todos los puntos de entrada tienen en cuenta las reglas de negocio y las validaciones en forma consistente.
- La integridad referencial de los datos en los múltiples repositorios puede ser violada a causa de la no existencia o el mal funcionamiento de los procesos de sincronización.
- Algunas aplicaciones pueden necesitar un subconjunto de datos que no están presentes físicamente en un solo repositorio. Para ello, la lógica de negocio tiene que ser aplicada a un subconjunto lógico de datos.
- El proceso de sincronización que existe entre dos repositorios permite que uno actúe como fachada del otro. En estos casos es mejor no alterar el mecanismo de sincronización. Este es un caso típico donde la replicación de datos es utilizada entre dos instancias de bases de datos separadas que utilizan la misma tecnología base.

Solución

Introducir una Capa de Agregación de Entidades que provea una representación lógica de las entidades a nivel empresarial. Los componentes de dicha capa encapsulan el acceso y la actualización de las instancias de las entidades en cada uno de los repositorios a través de conexiones físicas los mismos, como se muestra en la Figura 2.2.

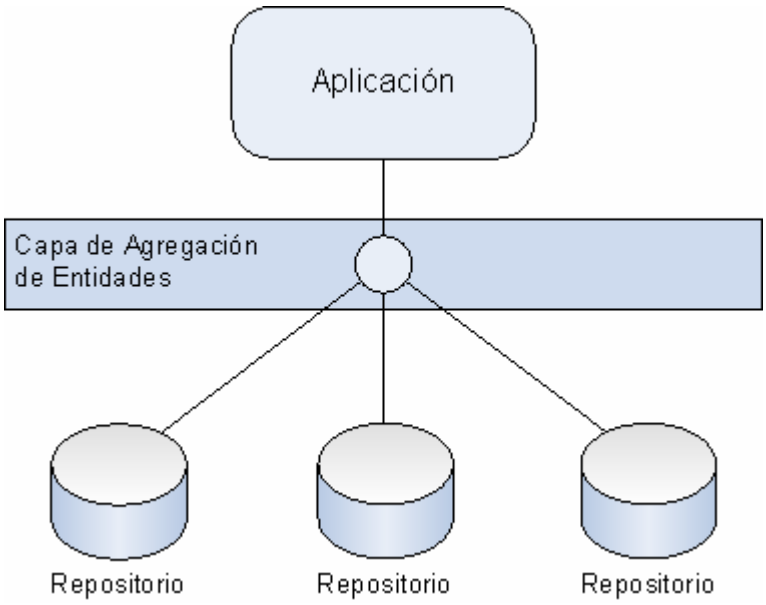


Figura 2.2 Capa de Agregación de Entidades

El establecimiento de una Agregación de Entidades comprende dos pasos:

- Definir una representación unificada y consistente de las entidades a nivel empresarial.
- Implementar una conexión bidireccional entre esta representación y sus instancias respectivas en cada uno de los repositorios.

Contexto Resultante

La Agregación de Entidades posee los beneficios y riesgos siguientes:

Beneficios

- *Consenso:* La Agregación de Entidades establece un consenso a través de las unidades funcionales y de negocios sobre la manera en que las entidades son representadas a nivel empresarial.
- *Vista única:* La Agregación de Entidades establece una vista única vista de las entidades de negocio centrales.
- *Acceso mejorado a la información:* Una vista única y completa de las entidades permite a las aplicaciones el acceso inmediato a toda la información. El acceso a la información no está limitado por los repositorios que la contienen.

- *Reducción de la disonancia semántica:* La Agregación de Entidades elimina las disonancias semánticas en las aplicaciones que usan los mismos elementos de datos de múltiples repositorios.
- *Localización Central:* La Agregación de Entidades constituye un punto central para la validación de los datos que van a ser introducidos en los repositorios.
- *Reducción del impacto por cambios:* La Agregación de Entidades reduce el impacto potencial por cambios en los repositorios. Dependiendo de la naturaleza de los cambios realizados, la Capa de Agregación de Entidades puede continuar satisfaciendo las necesidades de las aplicaciones durante los mismos.

Riesgos

- *Capa adicional:* La Agregación de Entidades introduce una capa adicional a la arquitectura que puede afectar el rendimiento global de la solución.
- *Consenso:* La Agregación de Entidades requiere un consenso a través de las unidades de negocio sobre la representación de las entidades, lo cual puede ser difícil o imposible de alcanzar.
- *Reingeniería de las aplicaciones:* Las aplicaciones existentes que estén fuertemente acopladas a un conjunto de repositorios tendrían que ser reprogramadas para acomodarlas a la nueva capa arquitectónica.

2.3.2 Capa de Integración de Procesos

Nombre

Integración de Procesos

Contexto

Se tienen múltiples sistemas diferentes, cada uno interviene en uno o más *procesos de negocio*[3]. Por ejemplo, el procesamiento de una Orden de Servicio es un proceso de negocio que requiere el de la participación del Sistema de Manejo de Clientes, el Sistema de Inventario, el Sistema de Transportación y uno o más sistemas financieros. El negocio

[3] En la literatura de hace referencia al concepto proceso de negocio para definir una función de negocio que abarca a varios sistemas. Se utiliza la palabra proceso para diferenciarla con las funciones de negocio que pertenecen al contexto de una aplicación en particular.

pudiera operar más de una forma más eficiente, si los sistemas estuvieran integrados de tal manera que los procesos pudieran ser completados automáticamente.

Problema

¿Cómo coordinar la ejecución a largo plazo de un proceso de negocio que abarca múltiples sistemas diferentes?

Fuerzas

Para solucionar este problema se deben balancear las consideraciones siguientes:

- Para la implementación del proceso, se puede delegar a cada sistema la responsabilidad de notificar al otro para ejecutar el paso siguiente. Esta variante crea dependencias entre los sistemas, las cuales hacen que los cambios en la secuencia de pasos sean más propensos al error y también limitan la capacidad de reutilizar los sistemas en múltiples procesos.
- El ciclo de mantenimiento de un proceso de negocio es diferente al ciclo de mantenimiento de las funciones de negocio dentro de cada aplicación. Por ejemplo, las funciones financieras como el cálculo de tasa de ventas no están sujetas a cambios frecuentes. En contraste, las estrategias de negocio y el *marketing* obligan a cambios frecuentes en la ejecución de los procesos de negocio con el objetivo de ganar en eficiencia y ajustarse a las necesidades del mercado.
- La mayoría de las funciones que están disponibles en las aplicaciones presentan un enfoque sincrónico, o sea, que el cliente tiene que esperar por el servidor hasta que este termine la ejecución. Este enfoque no es factible para los procesos de negocio que a menudo tardan días o semanas en ejecutarse, lo cual implica, que una aplicación pudiera emplear una cantidad significativa de tiempo esperando por otra.
- Un mayor tiempo de ejecución incrementa la probabilidad de fallos durante el proceso de negocio. Si ocurre un fallo, porciones de dicho proceso tendrían que ser revertidas para que el sistema retorne a un estado consistente.
- Los procesos de negocio toman un largo tiempo en ejecutarse, por ello es muy probable que surjan nuevas solicitudes mientras que la primera está todavía en proceso. Con el objetivo de mejorar los tiempos de respuesta, la solución debe

permitir manejar concurrentemente múltiples ejecuciones de un mismo proceso de negocio. Muchas aplicaciones no están diseñadas para este tipo de ejecución concurrente.

- La modelación de los procesos de negocios dentro de un componente de software puede resultar difícil si los procesos no son bien comprendidos o documentados. En muchos negocios, los usuarios toman decisiones basadas en su experiencia y no en reglas de negocio bien documentadas.

Solución

Definir un **modelo de proceso** que describa los pasos individuales que conforman un proceso de negocio. Crear un componente independiente (Gestor de Procesos) que permita interpretar concurrentemente múltiples instancias de este modelo. Dicho componente es el encargado de interactuar con las aplicaciones para llevar a cabo cada uno de los pasos individuales del proceso. La solución se muestra en la Figura 2.3.

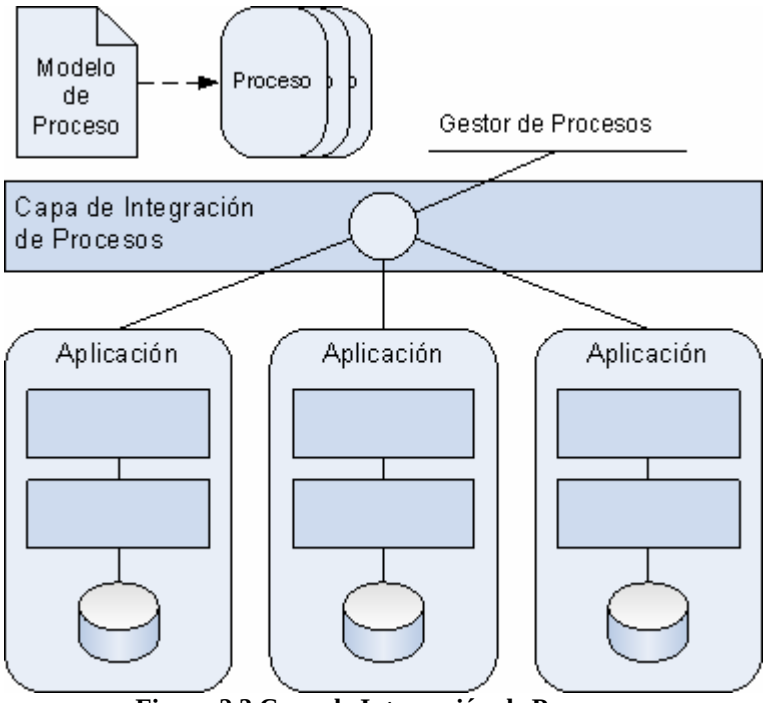


Figura 2.3 Capa de Integración de Procesos

Para cada solicitud de un proceso de negocio, el Gestor de Procesos crea una nueva instancia basado en el modelo del proceso. Esta instancia contiene el estado del proceso además de cualquier información que sea necesaria para la ejecución del mismo. Mediante

la creación de instancias individuales, el Gestor de Procesos posibilita la ejecución paralela de muchos procesos de negocio.

Después que una aplicación termina una función de negocio, el Gestor de Procesos indica qué función deberá ejecutarse a continuación basándose en el modelo del proceso y el estado de la instancia. De esta forma, cada aplicación puede operar individualmente sin conocimiento alguno de la secuencia de pasos.

El Gestor de Procesos mantiene el estado del proceso incluso si una aplicación no se encuentra disponible temporalmente. Como resultado, en vez de interrumpir la ejecución global del proceso, esta puede ser reanudada una vez que la aplicación vuelva a estar disponible.

El Gestor de Procesos interactúa con cada aplicación individual a través de la Integración de Datos, la Integración Funcional o la Integración de Presentación, dependiendo la naturaleza de las interacciones y la arquitectura de las aplicaciones. Por ello, la Integración de Procesos se asemeja a una aplicación compuesta construida encima de las funciones de negocio residentes en cada una de las aplicaciones.

El Gestor de Procesos permite hacer una separación clara entre la definición de los procesos (modelo del proceso) y la implementación de las funciones individuales de cada aplicación. Esta separación posibilita que las funciones puedan ser reutilizadas en múltiples modelos de proceso.

La interpretación de un modelo de proceso descrito en cierto lenguaje, hace que el Gestor de Procesos tenga un carácter general y sea reutilizable en múltiples dominios de aplicación. Los **lenguajes definición de proceso**, permiten la definición de ejecuciones paralelas, secuencias de pasos y puntos de decisión. A través de ellos, se pueden crear modelos que abstraigan las características internas del Gestor de Procesos y de las aplicaciones individuales.

Contexto Resultante

La Integración de Procesos posee los siguientes beneficios y riesgos:

Beneficios

- *Mantenibilidad:* La creación de una capa de integración de procesos permite definir y mantener los procesos de negocios independientemente de las implementaciones de las funciones individuales. La abstracción de las características intrínsecas de las aplicaciones y la introducción de un lenguaje de definición de procesos reduce en gran medida el conjunto de habilidades requeridas por el personal que mantiene los modelos de proceso.
- *Reusabilidad:* Al independizar las aplicaciones de los procesos, las funciones definidas en las mismas pueden ser reutilizadas en múltiples modelos de proceso.
- *Flexibilidad:* El Gestor de Procesos soporta una variedad de configuraciones que pueden ser difíciles de implementar en las aplicaciones tradicionales, por ejemplo, la ejecución paralela, los puntos de sincronización y la interrupción por tiempo entre otras. Estas configuraciones son particularmente difíciles de escalar una vez que sea requerido. El soporte de una gran cantidad de configuraciones hace del Gestor de Procesos una herramienta en extremo flexible y de fácil adaptación a muchos escenarios de negocio.
- *Capacidad de Reporte:* Como las instancias de los procesos son mantenidas centralmente por el Gestor de Procesos, es factible la extracción de información estadística. Esta información puede responder preguntas como: ¿cuál es el tiempo promedio que se tarda el llenado de una orden?, ¿cuántas órdenes se quedan pendientes a causa de insuficiencias en el inventario? ¿en que punto están detenidos determinados procesos?

Riesgos

- *Potenciales Cuellos de Botella:* El manejo de un gran número de instancias de procesos en un Gestor de Procesos central, puede representar un cuello de botella. No obstante, en la mayoría de los casos la ejecución paralela de muchas instancias del Gestor de Procesos puede aliviar esta situación sin renunciar a una configuración central.
- *Inclinación al Abuso:* Este riesgo constituye la cara oscura de la flexibilidad inherente a la Integración de Procesos. La Integración de Procesos puede ser usada para resolver casi cualquier problema de integración, por lo que algunas personas

consideran que todo problema de integración puede ser resuelto usando este patrón. El uso indiscriminado de la Integración de Procesos puede derivar en soluciones sobreestructuradas e ineficientes.

- *Complejidad:* Un Gestor de Procesos genérico es difícil de construir porque tiene que lidiar con fenómenos como la concurrencia, la validación y la recuperación ante los errores. Por estas razones siempre se recomienda el uso de un Gestor de Procesos profesional. En el caso de que esto no sea posible, se debe analizar cuidadosamente el costo de la construcción de un Gestor de Procesos propio en contraposición con el costo de la adaptación de los modelos de procesos de la empresa a un Gestor de Procesos comercial.

2.3.3 Capa de Integración de Portal

Contexto

Muchas operaciones requieren que los usuarios accedan a información que reside en varios sistemas. Por ejemplo, un cliente emite una orden de compras a través de un servicio de venta en línea. El representante necesita verificar la historia de pagos del cliente en el Sistema de Contabilidad (para cerciorarse que el cliente no tiene pagos atrasados), antes de aceptar la nueva orden. El constante cambio entre los dos sistemas hace que el trabajo del representante sea más tedioso y propenso al error.

Problema

¿Cómo pudieran los usuarios ejecutar eficientemente tareas que requieren el acceso a informaciones que residen en varios sistemas?

Fuerzas

- Aun hoy en día es muy común el copiado manual de la información de un sistema hacia otro. Este tipo de interacción a menudo se le denomina humorísticamente “la integración de la silla giratoria” ya que el usuario debe moverse entre varias pantallas tomando datos de una e introduciéndolos en la otra. Obviamente el tecleado manual de la información es ineficiente y muy propenso al error.
- La Integración de Procesos puede automatizar tareas que combinen datos y funciones de múltiples sistemas. No obstante, requiere una alta comprensión de la

secuencia de acciones para que puedan ser representadas mediante un modelo de proceso. En ocasiones esto es imposible, ya que los usuarios toman decisiones improvisadas a partir de las informaciones que ven.

- La Agregación de Entidades permite unir los datos de múltiples sistemas en un solo punto. Sin embargo, a veces aplicar una Agregación de Entidades no es posible o no es económicamente factible dadas las fuertes disonancias semánticas que pueden existir entre los sistemas.
- La lectura de datos desde una aplicación es generalmente más simple y mucho menos riesgosa que la actualización de los mismos. Esto se hace más evidente en los casos en los que se accede directamente a los repositorios saltando la lógica de validación que existe en la capa de negocios.

Solución

Crear una aplicación portal que visualice la información obtenida desde múltiples sistemas en una única interfaz como se muestra en la Figura 2.4. Los usuarios deben ser capaces de realizar sus tareas basándose en la información que es mostrada en este portal.

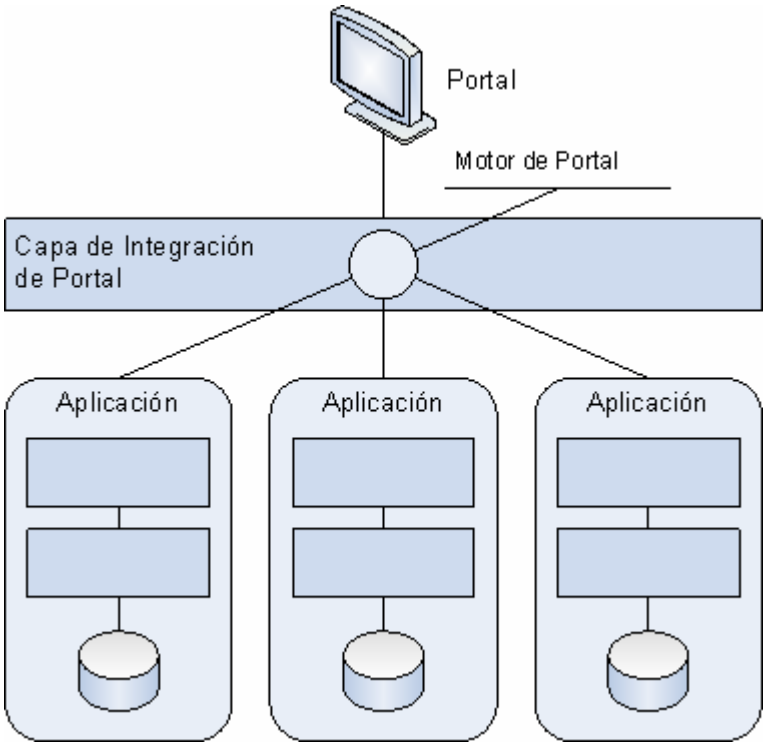


Figura 2.4 Capa de Integración de Portal

La Integración de Portal es comparativamente una forma simple de integración. Es popular pues es mucho más fácil de implementar y mucho menos intrusiva que otras formas más complejas de integración como la Integración de Procesos. En la Integración de Portal, un proceso de negocio conformado por una secuencia de tareas no tiene que estar representado mediante un modelo preciso, sino que esta representación reside en la mente del operario que ejecuta dichas tareas. La decisión de qué tareas ejecutar y cómo ejecutarlas es responsabilidad del operario. Este enfoque es menos eficiente que una Integración de Procesos pero compensa el efecto que producen las disonancias semánticas entre los sistemas al dar un grado mayor de flexibilidad a los operarios. En cuanto al tiempo de implementación, la Integración de Portal es mucho más rápida de implementar en la mayoría de los casos, por tanto es válido considerarla una solución intermedia en una transición a la Integración de Procesos.

Como se observa en la Figura 2.4, en el centro del modelo se ubica el Motor de Portal, el cual tiene la función de interactuar con las aplicaciones individuales utilizando Integración de Datos, Integración Funcional o Integración de Presentación. Como el propósito de la Integración de Portal es la visualización de la información, se prefieren formas de conexión relativamente simples como la Integración de Datos o la Integración de Presentación.

En la implementación de la Integración de Portal se puede elegir entre varias variantes que van desde las más simples hasta las más complejas. A continuación, se exponen una lista de ellas:

- *Solo Visualización:* Es la forma más simple de Integración de Portal. Se limita a visualizar la información en diferentes áreas de la misma pantalla. La información correspondiente a una aplicación está asociada a un área de la pantalla, que comúnmente se denomina **panel**. No existe ninguna otra funcionalidad adicional.
- *Post-Procesamiento Simple:* Además de presentar los datos de cada sistema individual, algunos portales agregan reglas simples que ayudan a los usuarios a tomar decisiones. Por ejemplo, si el Sistema de Pagos reporta un cliente con pagos atrasados, esta información pudiera aparecer resaltada en rojo en el tope de la

pantalla. Este tipo de funcionalidades tienden a convertir simples datos en informaciones muy útiles para los operarios.

- *Interacción con una Aplicación:* En esta variante además de visualizar los datos en diferentes áreas de la pantalla, se permite al operario introducir los datos en uno de los sistemas a la vez.
- *Interactividad Inter-Panel:* En muchos casos la información a ser visualizada en un panel depende de la selección del usuario en otro panel. Por ejemplo, el Sistema de Manejo de Clientes devuelve la lista de los clientes, lo cual es visualizado en un panel; cuando el operario selecciona un cliente, el historial de pagos de ese cliente es devuelto del Sistema de Pagos y visualizado en otro panel. Para la automatización de estos pasos, es necesaria una interacción básica entre los paneles del portal. De esta manera se acelera en gran medida el trabajo de los operarios sin la necesidad de integrar los dos sistemas completamente. Es de señalar que este tipo de portales requieren que los sistemas manejen una identidad común, como puede ser el identificador de usuario. Además, su desarrollo es difícil de poner a prueba y reutilizar ante los cambios.

Contexto Resultante

Beneficios

- *No Intrusivo:* La función primaria de la Integración de Portal es la visualización de la información. Esta funcionalidad puede ser agregada a las aplicaciones existentes sin necesidad de hacer cambios.
- *Velocidad de Implementación:* Agregar a un portal las aplicaciones existentes es una tarea que puede ser ejecutada en semanas e incluso días, mientras que una integración completa de las aplicaciones puede tardar muchos meses de trabajo. Por otro lado, muchos proveedores ofrecen plataformas para la Integración de Portal en combinación con una gran cantidad de paneles prefabricados los cuales se conectan a las aplicaciones y fuentes de datos más populares.
- *Flexibilidad:* Ya que el usuario es quien toma las decisiones, la Integración de Portal puede ser utilizada en situaciones en que las reglas de negocio no estén bien especificadas o no exista un consenso.

Riesgos

- *Ineficiencia:* La Integración de Portal se ajusta mejor a la automatización de tareas simples porque requiere de la interacción manual de los operarios. Debido a que la mayoría de los portales permiten la interacción con un sistema a la vez, las tareas deben ser ejecutadas en secuencia. En un proceso complejo con un alto volumen de aplicaciones, los operarios pueden desorientarse y emplear mucho tiempo en retomar el hilo del proceso. En estos casos la Integración de Procesos es una mejor solución pues la secuencia de pasos es controlada automáticamente por el Gestor de Procesos.
- *Propenso al Error:* Como las decisiones recaen completamente en los operarios, la flexibilidad cobra su precio al introducir el error humano. Por ello, la Integración de Portal no es recomendable para los escenarios que requieren una repetición estricta de un proceso de negocio definido, en estos casos es mejor usar Integración de Procesos.

2.4 Conexiones entre Sistemas

Cualquier solución de integración involucra la interconexión de sistemas. Conectar dos aplicaciones es un proceso complejo pues deben resolverse diversos retos que varían según el escenario de integración. Las conexiones dentro de la solución de integración constituyen un aspecto a ser analizado con vistas a proveer una capa de conexiones flexible, extensible y de fácil manejo, factores que influyen en la plataforma final.

A la hora de conectar las aplicaciones, los programadores se topan con una mezcla de arquitecturas técnicas, cada una de las cuales impone restricciones sobre las diferentes opciones de integración que se pueden implementar. Una revisión de los patrones de conexión es muy útil para seleccionar cada variante.

Al tratar de encontrar puntos de conexión en las aplicaciones, es preciso estudiar las arquitecturas de la misma. Entre ellas, el patrón prevaleciente lo constituyen las Aplicaciones Estratificadas en Capas [Trowbridge03], donde cada capa de la arquitectura representa una oportunidad de conexión para el especialista en integración. De aquí se desprenden los tres patrones de conexión expuestos: la Integración de Datos, la Integración

Funcional y la Integración de Presentación, que se corresponden con la conexión a la Capa de Datos, la Capa de Lógica del Negocio y la Capa de Presentación respectivamente.

2.4.1 Integración de Presentación

Contexto

Se tienes múltiples aplicaciones independientes, cada una posee una interfaz de usuario.

Problema

¿Como integrar dos sistemas de información que no están diseñados para trabajar en conjunto?

Fuerzas

- Cuando se integran dos sistemas de información se debe minimizar el cambio, pues cualquier cambio que se haga de un sistema existente representa un riesgo y necesita de pruebas extensivas para asegurarse de que todo funciona correctamente.
- Puede ocurrir que todas las capas de una aplicación no sean accesibles desde el exterior. Este es el caso de muchas aplicaciones Web en que la Capa de Negocio y la Capa de Datos están protegidas por un cortafuegos, al usuario sólo se le permite acceder a la Capa de Presentación.
- Algunas aplicaciones tienen poca o ninguna separación entre las capas. A veces la Capa de Presentación esta fusionada con la lógica del negocio, por ejemplo en las aplicaciones de dos capas [Rodríguez03] los controles acceden directamente a los datos. En otras ocasiones la Capa de Datos y la lógica del negocio están fusionadas dentro del gestor de datos como en el caso de los antiguos sistemas montados en *mainframes*.
- Algunas Capas de Negocio están implementadas en lenguajes y tecnologías que no permiten que sean accedidos de manera remota.
- Acceder directamente a los datos, saltándose la lógica de validación, puede corromper los datos.

Solución

Acceder a la funcionalidad de la aplicación al nivel de la Capa de Presentación simulando la entrada y leyendo los datos a través de la interfaz de usuario. El esquema de la solución se muestra en la Figura 2.5.

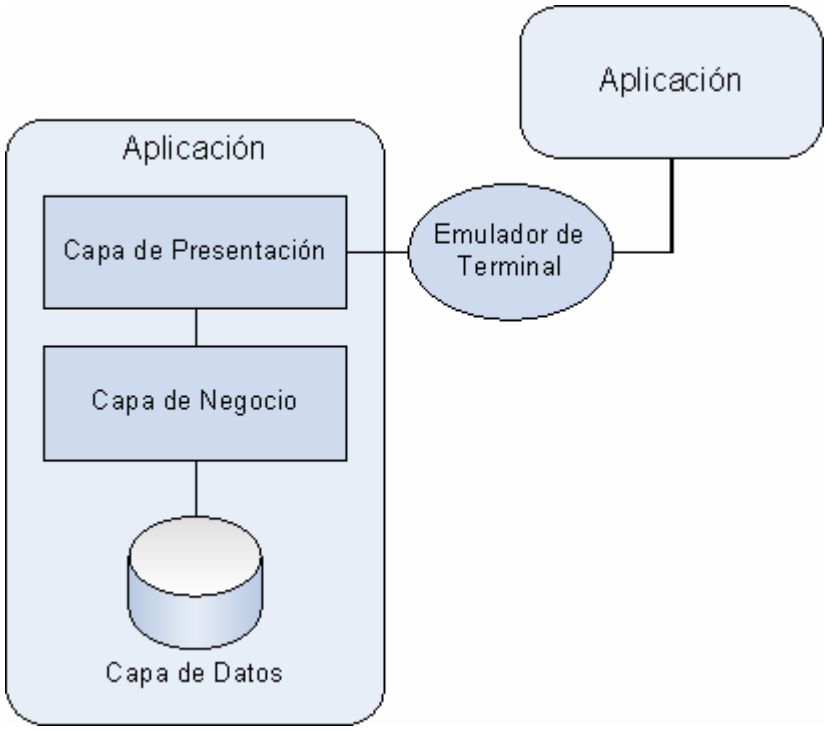


Figura 2.5 Integración de Presentación

Para ello es necesario construir un módulo de emulación de la terminal de usuario. La construcción de dicho módulo puede ser tan simple como enviar comandos de texto a una terminal de comandos y capturar la salida, o tan compleja como tratar de integrarse a una interfaz visual moderna. Por ello se debe analizar con cuidado las características de la interfaz a la que se pretende conectar.

Contexto Resultante

Beneficios

- *Bajo riesgo:* El riesgo de efectuar operaciones no válidas y corromper los datos es prácticamente nulo, ya que toda la lógica de validación contenida en las tres capas se ejecuta. Esto es muy importante en aplicaciones antiguas, en las cuales los mecanismos están poco documentados y son de difícil comprensión.

- *No agresivo:* Toda la interacción con la aplicación ocurre de la misma manera que ocurriría si fuese un usuario real. No hay necesidad de operar ningún cambio a las aplicaciones existentes.
- *Funciona en aplicaciones monolíticas:* Aun en aplicaciones en que no existe una separación clara entre las capas es posible utilizar este modelo. También es aplicable a escenarios donde las capas de negocio y de datos no son accesibles por alguna razón.

Riesgos

- *Fragilidad:* La interfaz de usuario está sujeta a cambios frecuentes entre las versiones de la aplicación. El costo del constante mantenimiento que necesita la solución de integración puede ser significativo.
- *Acceso limitado a los datos:* La Integración de Presentación solo puede acceder a los datos que se muestran en la interfaz de usuario. Las informaciones internas que son muy útiles para la integración, como es el caso de las llaves primarias de las entidades, casi nunca se muestran en la interfaz.
- *Información poco estructurada:* En la mayoría de los casos la capa de presentación formatea los datos para hacerlos presentables al ojo humano. Los metadatos no se muestran en la interfaz y se pierde el nexo entre los datos y las entidades lógicas de la aplicación. En estos casos, es preciso programar una lógica al módulo de emulación que agregue semántica a los datos capturados.
- *Ineficiencia:* En la Integración de Presentación se ejecuta el código asociado a la representación visual de la información. Posteriormente, es necesario ejecutar otro código para revertir esta representación en datos útiles a las aplicaciones. La representación visual solo tiene sentido en la interacción con los humanos. En el caso de una aplicación esto es innecesario.
- *Lentitud:* La lentitud es causa del complejo proceso de reconocimiento de la información. Por ejemplo, en algunas ocasiones la información que es presentada al ojo humano se encuentra resumida y paginada para ajustarla a un espacio visual, esto requiere un trabajo extra del módulo de emulación para recorrer varias páginas y extraer un conjunto coherente de datos.

- *Complejidad:* Simular la actividad de un usuario humano es un proceso que puede derivar en una complejidad tal, que no sea factible su implementación. Esto se debe a que los pasos que son ejecutados instintivamente por cualquier ser humano tienen que ser programados implicando una gran cantidad de código bastante complicado.

2.4.2 Integración de Datos

Contexto

La información concerniente a una empresa está distribuida sobre una gran variedad de sistemas de almacenamiento de datos, que van desde los más completos gestores de datos hasta un simple fichero.

Problema

¿Cómo integrar sistemas de información que no están diseñados para trabajar en conjunto?

Fuerzas

- La mayoría de las empresas contienen sistemas que no están diseñados para trabajar en conjunto. Estos sistemas están fabricados y soportados sobre una diversidad de tecnologías que, en la mayoría de los casos, son incompatibles.
- La mayoría de las aplicaciones siguen el patrón de tres capas: una capa de datos, una de lógica del negocio y una de presentación.
- A la hora de integrar múltiples sistemas se trata de ser lo menos agresivo posible. Cualquier cambio que se opere en un sistema en explotación constituye un riesgo, por lo que es sabio buscar una solución que satisfaga las necesidades y que a su vez minimice las alteraciones.
- Siempre es recomendable aislar las estructuras de datos internas de la aplicación, o sea, que el cambio de dichas estructuras de datos no afecte a otras aplicaciones. Sin el aislamiento, un pequeño cambio en una de las aplicaciones se propagaría a las aplicaciones dependientes, obligando a los desarrolladores a cambiarlas y así sucesivamente.
- La lectura de los datos de un sistema, en muchas ocasiones requiere muy poca o ninguna lógica de validación. En estos casos acceder directamente a la Capa de Datos es más eficiente que hacerlo a través de la Capa de Negocio.

- Muchas aplicaciones mezclan la capa de presentación y la capa de lógica del negocio, lo que provoca que esta última no sea accesible externamente. En otros casos la lógica del negocio está implementada en lenguajes o tecnologías que no soportan el acceso remoto. Ambos escenarios limitan la capacidad de una Integración Funcional.
- Al actualizar los datos de otra aplicación, se debe aprovechar la lógica de negocio para validar y chequear la integridad referencial. En estos casos es mejor considerar una Integración Funcional.
- El acceso directo a los datos de una aplicación puede violar las políticas de seguridad que comúnmente están implementadas en la Capa de Negocio.
- Se debe considerar la existencia de herramientas comerciales para la integración que disminuyan los riesgos y el costo en tiempo y recursos.

Solución

Integrar las aplicaciones en la Capa de Datos permitiendo que los datos de una aplicación (la fuente) puedan ser accedidos por otras aplicaciones (los destinos), como se muestra en la Figura 2.6.

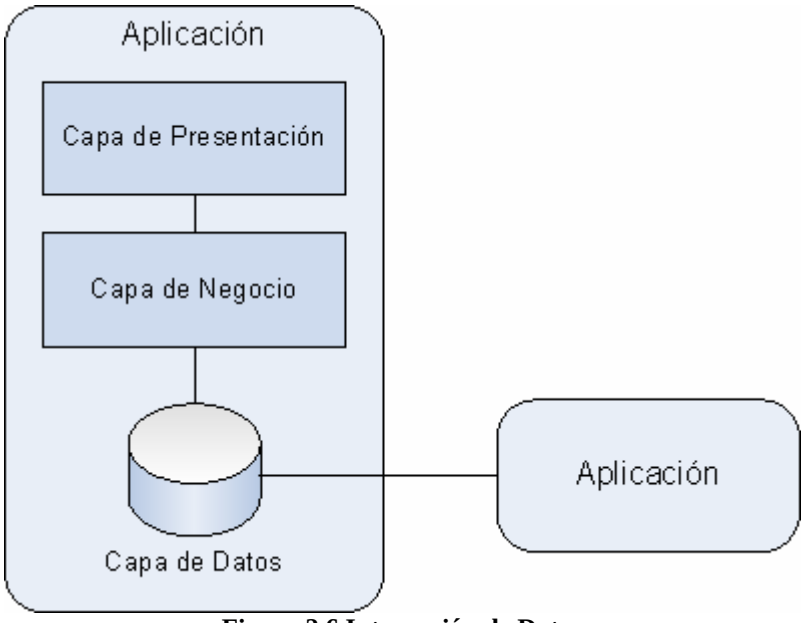


Figura 2.6 Integración de Datos

Una vez que se opta por la Integración de Datos puede considerarse el uso de uno o varios de los siguientes refinamientos del patrón:

- *Bases de Datos Compartidas:* Todas las aplicaciones que se están integrando acceden a los datos de la misma base de datos.
- *Mantener Copias de los Datos:* Se mantiene una copia separada de los datos de la aplicación para las otras aplicaciones, esta copia está orientada a lectura y potencialmente a la actualización.
- *Transferencia de Ficheros:* Extraer los datos de una aplicación en un fichero que se pueda importar en otras aplicaciones.

La elección de cual de los refinamientos es el adecuando debe balancear los siguientes factores:

- Tolerancia a la desactualización.
- Rendimiento.
- Complejidad.
- Infraestructura de la plataforma y disponibilidad de herramientas.

A partir del momento en que se extraen datos desde una base de datos, existe una probabilidad de que estos ya estén desactualizados. Por tanto, cuando se van a utilizar para una operación de actualización, necesariamente hay a que pasar por un proceso de resolución de conflictos. Si la lógica de resolución de conflictos es simple y además se pueden permitir tiempos relativamente largos con datos desactualizados, entonces la Transferencia de Ficheros es una buena opción. En cambio, si la lógica de resolución de conflictos es más compleja y existe poca tolerancia a la desactualización, es necesario considerar Bases de Datos Compartidas o Mantener Copias de los Datos. La elección entre ambos casi siempre es decidida en base al rendimiento. Si todos los datos conviven en el mismo gestor, por ejemplo un cluster de datos, se puede optar por las Bases de Datos Compartidas. Al Mantener Copias de los Datos se mejora el rendimiento promedio y se permite un número mucho mayor de aplicaciones conectadas, pero necesariamente hay que lidiar con una lógica de sincronización. En este caso, es preciso revisar las facilidades de sincronización que brinde la arquitectura o, en última instancia, implementar un mecanismo propio.

Al implementar cualquiera de estos mecanismos deben analizarse los aspectos siguientes:

- *Latencia:* Algunas formas de integración de datos introducen un mayor retardo para que las actualizaciones sean visibles en las aplicaciones. Por ejemplo, en un patrón de Replicación de Datos [Teale03] los datos actualizados son extraídos de un sistema, transportados a través de la red, transformados, para luego ser insertados en la base de datos de destino. Esta demora provoca que las aplicaciones estén consumiendo datos desactualizados por un largo tiempo.
- *Introducción versus Extracción:* Una aplicación puede optar por extraer los datos desde las fuente externas o en cambio, dejar que estos le sean introducidos. El enfoque de la Extracción es generalmente el menos agresivo, mientras que la Introducción tiende a disminuir la Latencia.
- *Relaciones Amo-Esclavo:* Cuando solamente existe una aplicación encargada de la actualización de los datos, la propagación de los cambios puede lograrse fácilmente. Sin embargo, si múltiples aplicaciones pueden actualizar los datos concurrentemente pueden aparecer problemas de sincronización. En este caso, un análisis más detallado se hace en el patrón *Replicación Amo-Amo* [Teale03].
- *Lógica de Sincronización versus Latencia:* Para las aplicaciones que están separadas geográficamente el acceder a la misma base de datos a través de la red puede provocar una latencia excesiva. Para contrarrestar este efecto se pueden mantener copia de los datos en diversos puntos, sin embargo esto conlleva a un incremento en la complejidad de los mecanismos de sincronización y replicación de los datos.

Contexto Resultante

Después que se opta por la Integración de Datos, debe seleccionarse un patrón particular de Integración de Datos de los que se mencionan a continuación:

- Bases de Datos Compartidas
- Mantener Copias de los Datos.
- Transferencia de Ficheros

Beneficios

- *No es agresivo:* La mayoría de los gestores permiten el acceso concurrente de múltiples usuarios de manera transaccional, por lo que fácilmente se puede

establecer una nueva conexión a los efectos de la integración. Por otra parte, la exportación e importación de ficheros en diversos formatos es común en casi todas las aplicaciones. Esto hace de la Integración de Datos una opción natural a la hora de integrar aplicaciones cerradas o difíciles de modificar.

- *Mayor ancho de banda:* Las conexiones directas a bases de datos y los ficheros están diseñados para soportar el trabajo con grandes volúmenes de datos de manera eficiente. Esto es especialmente útil cuando se requieren operaciones que afectan a múltiples entidades.
- *Acceso a los datos crudos:* En algunos escenarios de integración, el acceso a los datos crudos es más efectivo desde la Capa de Datos que desde otras capas donde los datos ya han sido transformados para su visualización. Por ejemplo, el acceso directo a los datos permite recuperar las llaves primarias que identifican a cada una de las entidades. Estás llaves, que muchas veces son ocultadas por la Capa de Negocio, son vitales para una solución robusta de integración.
- *Acceso a los metadatos:* La mayoría de los gestores comerciales de datos permiten el acceso a metadatos como: nombre de los datos, tipo, relaciones entre las entidades, etc. Constar con metadatos simplifica la lógica de las operaciones con los datos y permite la implementación de algoritmos más potentes.
- *Soporte en cuanto a herramientas:* El mercado de las bases de datos es uno de los más desarrollados dentro de la producción de software y consta con una gran gama de herramientas y tecnologías de alto nivel que permiten atacar los más diversos problemas.

Riesgos

- *No publicación de los esquemas:* Los esquemas de las bases de datos de la mayoría de los sistemas no son publicados. Los fabricantes con frecuencia se reservan el derecho de hacer cambios en la estructura de la base de datos que afecten la compatibilidad de una versión a otra. Esto atenta contra la mantenibilidad de las soluciones que usan Integración de Datos. Otra consecuencia derivada es que los esquemas de las bases de datos no están documentados, obligando a los desarrolladores a consultar a los productores o a la ingeniería inversa.

- *No existe encapsulamiento:* El acceso a los datos crudos puede ser una ventaja en ocasiones, pero permite muy poco o ningún encapsulamiento de la funcionalidad de la aplicación. Esto constituye un incremento de la complejidad que además trae aparejado operaciones de transformación para limar las diferencias estructurales y semánticas.
- *Semántica simplificada:* Como su nombre indica, la Integración de Datos integra solamente los datos, no permite la utilización de la lógica de negocio de las aplicaciones.
- *Diferentes paradigmas de almacenamiento:* Aunque los gestores relacionales son el soporte más popular de datos, existen otros soportes ampliamente difundidos como los directorios, ficheros planos, XML, etc. Esto hace que se necesite una lógica adicional para la transformación entre los diferentes paradigmas.
- *Disonancia semántica:* La disonancia semántica es un problema común en la integración de datos. La falta de una documentación precisa agrava aun más el problema, pues las disonancias pasan inadvertidas por los programadores lo que provocaría errores en el futuro.

2.4.3 Integración Funcional

Contexto

Los sistemas de información de una empresa contienen una variedad de aplicaciones que proveen diferentes niveles de interacción. Las funcionalidades incluidas en estos sistemas ofrecen una variedad de capacidades y niveles de acceso que van desde funciones indocumentadas en aplicaciones aisladas hasta aplicaciones compuestas completamente desarrolladas. Además, el mecanismo de invocación de las funciones puede variar de una aplicación a otra.

Problema

¿Cómo integrar sistemas de información que no están diseñados para trabajar en conjunto?

Fuerzas

- La mayoría de las empresas contienen sistemas que nunca fueron diseñados para trabajar en conjunto. Como los sistemas de información tienden a ser muy variables

en cuando a su arquitectura técnica, al contrario de una tecnología homogénea, lo más común es encontrase una mezcla de tecnologías y arquitecturas incompatibles.

- La mayoría de las aplicaciones siguen el patrón de tres capas: una capa de datos, una de lógica del negocio y una de presentación.
- La mayoría de los sistemas desarrollados durante la última década proveen interfaces de programación estables y documentadas que permiten el acceso a la funcionalidad de negocio incorporada en la aplicación. Los productores proveen estas API[4] para soportar la integración con software externo.
- Las interfaces funcionales que proveen las aplicaciones abstraen la representación subyacente de los datos, por lo que son más estables y con frecuencia se mantienen a través de las diferentes versiones del software.
- El acceso a una función que reside en otra aplicación puede ser visto como una extensión del modelo. La semántica involucrada en el acceso a una función remota es abstraída por muchas tecnologías en forma similar a una llamada local. Esto permite, a la vista de los programadores, una forma mucho más simple y natural de integración.
- Actualizar directamente los datos de otra aplicación al nivel de la capa de datos implica saltarse toda la lógica de validación incorporada en la capa de negocio. Como resultado, el riesgo de corromper los datos es alto. Esto no sucede si se usan las funciones que provee la aplicación.
- El intercambio de mensajes que contengan no solamente datos sino también operaciones establece una comunicación mucho más poderosa entre los sistemas. Por ejemplo, en vez de enviar solamente los datos de un cliente hacia el Sistema de Clientes, se podría pedir al sistema que valide los datos y posteriormente confirme su actualización. Formas mucho más poderosas de comunicación entre dos sistemas permiten incluso que un sistema describa dinámicamente a través de dicha "conversación" las funcionalidades o servicios que oferta.
- Muchas de las interfaces de programación no están disponibles de manera remota a no ser que estén construidas encima de una tecnología que permita acceso remoto

[4] Del inglés Aplication Program Interface.

como: Microsoft .Net Remoting, Distributed Common Object Model (DCOM) o Commom Object Request Broker Architecture (CORBA).

Solución

Integrar las aplicaciones en la capa de negocio haciendo que algunas funciones de negocio de una aplicación (la fuente) puedan ser accedidas por otras aplicaciones (los destinos). El esquema de la solución se muestra en la Figura 2.7

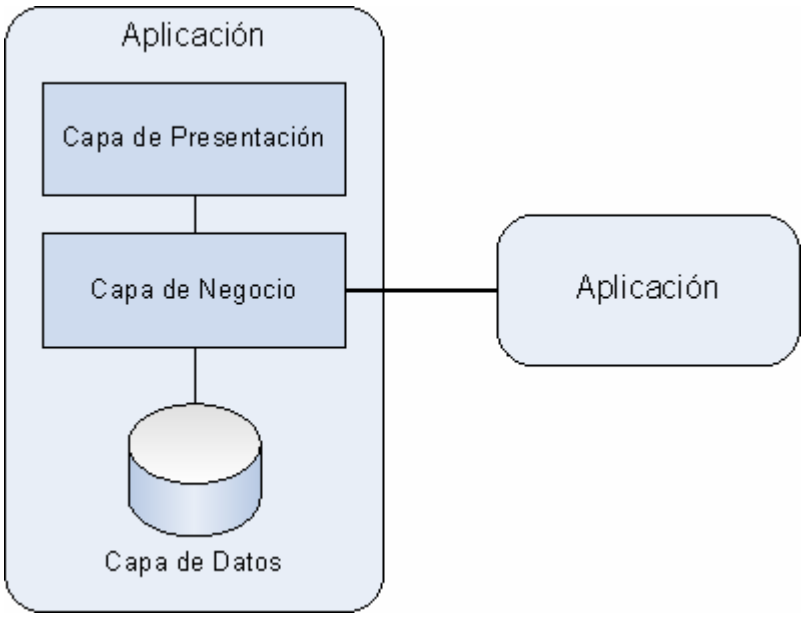


Figura 2.7 Integración Funcional

Para que una aplicación externa se integre con la aplicación fuente deben cumplirse las condiciones siguientes:

1. Las funciones de negocio deseadas deben estar disponibles dentro de la aplicación fuente.
2. Dichas funciones deben ser accesibles remotamente.

Si la función deseada no está disponible en la aplicación fuente, se puede modificar la aplicación para que lo esté. Si no es factible la modificación, se puede crear la función fuera de la aplicación y luego comunicarse con la aplicación a través de otra forma de integración como la Integración de Datos. Este enfoque tiene muy pocos efectos secundarios y es factible para la mayoría de las aplicaciones.

Muchas aplicaciones solo exponen sus funciones de negocio como una API local, que es dependiente de un lenguaje de programación específico como C# o C++. En estos casos, es necesario crear un Adaptador [Patternshare05] que traduzca los mensajes de otras aplicaciones en llamadas la API local. Asimismo los resultados de la llamada son traducidos en mensajes de respuesta. En la mayoría de los casos, estos adaptadores son lo suficientemente genéricos como para que soporten una variedad de funciones y no exista la necesidad de crear uno específico para cada función que se pretenda hacer accesible externamente.

Contexto Resultante

Después que se ha decidido usar Integración Funcional, se pueden considerar uno de los refinamientos siguientes:

- Integración mediante Objetos Distribuidos.
- Integración mediante una Capa Intermedia Orientada a Mensajes.
- Integración Orientada a Servicios.

Integración mediante Objetos Distribuidos.

La Integración mediante Objetos Distribuidos es también conocida como la "colaboración basada en instancias" porque extiende el modelo de objetos hacia las soluciones distribuidas. Los objetos dentro de la aplicación interactúan con los objetos de la aplicación remota de la misma forma que lo harían con objetos locales. Los clientes interactúan con una instancia específica del objeto en el servidor y con frecuencia son los encargados de manejar el tiempo de vida de dichas instancias. Este tipo de interacción se presenta de forma natural ante los desarrolladores, pero internamente implica una interacción compleja entre componentes fuertemente acoplados. El acoplamiento fuerte no es un problema mientras los componentes sean parte de una misma aplicación distribuida, sin embargo, no es una buena opción al integrar aplicaciones heterogéneas.

Como ejemplo de tecnologías que permiten la Integración mediante Objetos Distribuidos se puede citar a .Net Remoting, COM+ y CORBA. Para una mayor información se puede revisar el patrón de Invocación de Procedimientos Remotos [Hohpe04], el capítulo de Sistemas Distribuidos expuesto por Townbridge [Townbridge03].

Integración mediante una Capa Intermedia Orientada a Mensajes.

La Integración mediante una Capa Intermedia Orientada a Mensajes permite la comunicación mediante el paso mensajes asincrónicos entre las aplicaciones. Los mensajes viajan a través de una arquitectura de mensajería, más específicamente, de arquitecturas de colas de mensajes. Estas arquitecturas posibilitan una comunicación confiable a través de redes poco confiables, permiten lidiar con largos tiempos de espera y soportan para una mayor carga de usuarios. Algunas de las tecnologías más utilizadas en este sentido son MSMQ y JMS.

La implementación de una Capa Intermedia tiene como objetivo conectar un modelo de negocios implementado sincrónicamente a una arquitectura de mensajería implementada asincrónicamente. Esto contribuye a disminuir el acoplamiento y favorece la integración entre las aplicaciones, que el lo adelante se comunicarán a través del sistema de mensajería. Para una mayor información sobre los beneficios este tipo de integración se puede consultar el patrón de Mensajería [Hohpe04].

La construcción de la Capa Intermedia puede ser una tarea costosa, sin embargo, su principal problema, es el alto grado acoplamiento de los sistemas con la arquitectura de mensajería.

Integración Orientada a Servicios

La Integración Orientada a Servicios, como su nombre lo indica, es la conexión a los sistemas a través de sus **servicios**. Este tipo de integración está diseñada para resolver los dos problemas fundamentales de los patrones anteriores: el acoplamiento y la interoperabilidad.

Los servicios parten del establecimiento de un **contrato** que rige la comunicación entre ambas partes. El contrato que separa la semántica del servicio, de las especificidades de su implementación, este enfoque revolucionario de la tecnología, ha provocado reflexiones en la teoría de programación y ya se habla de conceptos como Aplicaciones Orientadas a Servicios y Arquitecturas Orientadas a Servicios.

La comunicación con los servicios ocurre mediante el paso de mensajes, con lo que se obtienen todas las ventajas de la mensajería [Hohpe04]. Además, están preparados para soportar ambos modelos de comunicación: sincrónica y asincrónica.

Las tecnologías para el desarrollo de servicios, aunque jóvenes todavía, ya constituye la opción principal de la integración. Esto ha hecho que los productores desarrollen soluciones que exploten las capacidades de las tecnologías anteriores usando el nuevo enfoque de servicio.

Beneficios

- *Flexibilidad:* La Integración Funcional es sumamente flexible. La abstracción de la comunicación en la forma de una invocación a una función puede ser usada en cualquiera de las capas de integración.
- *Encapsulamiento:* El establecimiento de una interfaz funcional aísla los usuarios de la implementación interna de la aplicación y la estructura interna de los datos, permitiendo a las aplicaciones evolucionar sin afectar a las aplicaciones dependientes.
- *Robustez:* La integración a nivel de la capa de negocios permite la ejecución de toda la lógica de validación implementada en esta capa, disminuyendo la probabilidad de que los errores puedan corromper los datos.
- *Familiaridad con el modelo de programación:* La Integración Funcional provee un modelo de programación que está más difundido entre los desarrolladores. Existe un mayor grado de familiarización con las tecnologías y metodologías que se usan en este tipo de integración.

Riesgos

- *Acoplamiento fuerte:* Cuando una aplicación envía comandos directamente hacia otra, ambas poseen un acoplamiento más fuerte que si los mensajes son publicados a un medio común, pues en el primer caso la fuente debe mantener la información sobre donde se ubican los puntos de conexión de los destinos para poder comunicarse. El efecto del acoplamiento puede ser mitigado a través del patrón Broker o el Bus [Townbrigde04].

- *Requiere la exposición de la capa de negocio:* La integración funcional necesita que las aplicaciones expongan una interfaz apropiada de la capa de negocio. Esto es una tarea sencilla en la mayoría de las aplicaciones modernas que manejan el concepto de capas, pero en aplicaciones monolíticas antiguas y sobre todo en las primeras tecnologías *Web* donde la capa de negocio está fusionada con la capa de presentación, puede no ser factible.
- *Limitado a las funciones disponibles:* En la mayoría de los casos, la Integración Funcional está limitada a las funciones implementadas en la aplicación. La extensión de la aplicación con el objetivo de agregar nuevas funcionalidades puede ser difícil si la aplicación no está preparada para ello. En estos caso se puede considerar la alternativa de implementar las nuevas funciones de manera externa.
- *Ineficiente con grandes conjuntos de datos:* Las interfaces funcionales existentes casi siempre están pensadas para la ejecución de funciones individuales de granularidad fina. Esta característica las hace ineficientes para la transmisión de grandes conjuntos de datos, los cuales necesitarían de una gran cantidad de llamadas.
- *Específicas de un lenguaje de programación:* Muchas interfaces funcionales están atadas a un lenguaje de programación o a una tecnología específica, pues mientras más fuerte es la semántica de las conversaciones, más detallados deben ser los contratos en cuanto a la representación de los nombres de los procedimientos, tipos de datos complejos, valores de retorno, excepciones y otros elementos.

2.4.4 Integración Orientada a Servicios

Contexto

Una vez que se ha decidido utilizar la Integración Funcional para conectar dos sistemas de información, es necesario que esto ocurra entre diferentes arquitecturas técnicas. Adicionalmente se observa que la conexión entre los sistemas no es confiable.

Problema

¿Cómo integrar estas aplicaciones a nivel de la capa de negocio?

Fuerzas

- *Los límites de la máquina son importantes:* La idea de tomar una interfaz local y extenderla fuera de los límites físicos de la máquina creando una ilusión de transparencia, es engañosa. Aunque es cierto que las representaciones locales de los objetos remotos poseen la misma interfaz, existen grandes diferencias en el proceso de llamada. Desde el punto de vista del cliente, las implementaciones remotas están sujetas a la latencia y los fallos de las redes, problema que no existe en una llamada local. Esto hace que sea necesario escribir una cantidad significativa de código para la manipulación de los errores con vistas a disminuir el impacto de los mismos.
- *El acoplamiento afecta la interoperabilidad:* La integración a nivel funcional implica cierta funcionalidad compartida entre dos aplicaciones, lo que exige ciertos niveles de acoplamiento. Existen diversas clases de acoplamiento entre las cuales se incluyen las siguientes:
- *Acoplamiento temporal:* Esta clase de acoplamiento ocurre durante la llamada a una función remota. El sistema que hace la solicitud de cierta funcionalidad depende del sistema que la satisface durante un intervalo de tiempo. Si la llamada es sincrónica el primer sistema espera por la respuesta del segundo para continuar su ejecución. Si la llamada es asincrónica el sistema puede continuar funcionando después que hace la solicitud y cuando llegue entonces procesar la misma. En términos de tiempo, el acoplamiento de la comunicación asincrónica es más débil que el de la sincrónica. A su vez, esta última es menos compleja que la primera.
- *Acoplamiento del sistema de tipos:* Los nombres de los tipos de datos y la representación binaria de los valores de los mismos, varían de una plataforma a otra. Para que dos sistemas puedan comunicarse exitosamente deben acordar el uso de tipos de datos comunes. La mayoría de las plataformas no concuerdan en cuanto a sus tipos de datos.
- *Acoplamiento de dependencia:* El acoplamiento de dependencia ocurre cuando un ejecutable depende de otro. Si el primero no puede resolver sus dependencias con el segundo, entonces falla. En este sentido, ambos ejecutables están acoplados.

- Para poder integrar con éxito dos aplicaciones debe tenerse un sistema de tipos de datos común, garantizar que se cumplan las dependencias de ejecución y analizar los efectos del acoplamiento temporal.
- *El acoplamiento al Lenguaje de Definición de Interfaces:* Muchas soluciones de integración resuelven el problema del acoplamiento de tipos usando tipos definidos en un lenguaje intermedio, este lenguaje recibe el nombre de Lenguaje de Definición de Interfaces. Ello requiere que cierta infraestructura para la interpretación y la conversión de los tipos esté instalada en ambos sistemas. Por ejemplo, CORBA define sus tipos interoperables en un lenguaje llamado CORBA IDL (CORBA Interface Definition Language), este lenguaje requiere de una estructura llamada ORB (Object Request Broker) que transforma los tipos de datos en un sentido y en el otro. Aunque el establecimiento de un Lenguaje de Definición de Interfaces resuelve el problema de interoperar sistemas de tipos diferentes, las aplicaciones en ambos extremos de la comunicación están acopladas a las estructuras que hacen la conversión de los tipos.
- *El acoplamiento en las capas intermedias orientadas a mensajes:* Cuando se usa una tecnología propietaria de colas de mensajes para la capa intermedia, cada aplicación debe tener dicha tecnología instalada. En este sentido las aplicaciones están acopladas a una tecnología que puede no estar disponible para todas las plataformas o sistemas operativos. Esto restringe, en gran medida, el arco de tecnologías que se pueden usar en la solución.
- *Sistema de tipos portable:* Casi la totalidad de los sistemas pueden procesar XML, ya que la única primitiva que se requiere para ello es el concepto de cadena de caracteres. Después que se maneja este concepto, cualquier XML puede ser procesado usando reglas bien definidas en su especificación, en particular los *XMLSchema*. El *XMLSchema* provee un sistema de tipos altamente portable para los datos expresados en XML con soporte en casi todas las plataformas y cuenta con la aceptación de los productores de tecnologías en todo el mundo.
- *Contratos:* El establecimiento de un contrato explícito que establezca las reglas para la comunicación entre dos partes ha probado su eficiencia.

Solución

Preparar las aplicaciones para que sean capaces de proveer y consumir Servicios Web basados en XML. Utilizar WSDL (Web Services Definition Language) para definir los contratos de dichos servicios. Utilizar XMLSchemas para definir los tipos que participan en la comunicación. Asegurar la interoperabilidad mediante el uso de la familia de especificaciones para resolver las necesidades de comunicación de los Web Services (WS-Addresing, WS-Trust, WS-SecureConsersation, etc) antes de optar por una solución propia.

Una definición formal de Servicio Web acorde con lo planteado por el patrón sería: "Un Servicio Web es un software diseñado para soportar una interacción interoperable máquina a máquina a través de la red. Posee una interfaz descrita en un formato procesable por la máquina (específicamente WSDL). La interacción de otros sistema con el Servicio Web ocurre en la manera descrita por esta interfaz y a través de mensajes SOAP, usualmente transportados por HTTP con serialización en XML en conjunción con otros estándares relacionados con la Web." [W3C04]

Los Servicios Web cuentan de tres partes como se muestra en la Figura 2.8. La Implementación del Servicio, que encapsula la capa de negocio incrementando la granularidad al nivel necesario; la Interfaz de Servicio (Service Interface [Trowbridge03]), que expone dicha funcionalidad para que pueda ser accedida mediante determinado protocolo de comunicación y la Puerta de Servicio (Service Gateway [Trowbridge03]), que encapsula en el cliente los detalles de la comunicación.

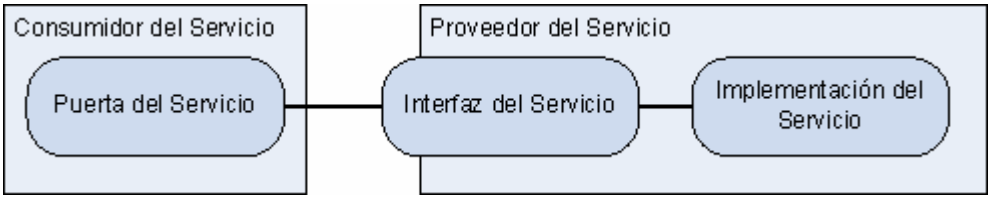


Figura 2.8 Estructura de un Servicio Web

En un Servicio Web los límites de la máquina están definidos explícitamente por la interfaz, permitiendo a los clientes manejar conceptos como la latencia y los fallos.

Los Servicios Web están teóricamente concebidos con independencia de transporte. Esto significa que pueden utilizar cualquier transporte para la transmisión de sus mensajes,

incluyendo una arquitectura de colas de mensajes. Lo anterior fusiona las ventajas de fiabilidad de las arquitecturas de colas de mensajes con las ventajas de interoperabilidad de los Servicios Web.

Los Servicios Web utilizan XMLSchema y WSLD como lenguaje de definición de tipos y de interfaces respectivamente, ambos lenguajes son altamente portables lo que soluciona el problema del acoplamiento al sistema de tipos.

Los contratos de los Servicios Web quedan definidos explícitamente a través de XMLSchema, WSDL y WS-Policy, donde se especifican los elementos para interoperar con los servicios. De modo que cualquier sistema interesado sólo tiene que adjuntarse a dicho contrato.

Los Servicios Web permiten la comunicación tanto sincrónica como asincrónica. Se deja en manos del desarrollador escoger cual es las más apropiada según sus prioridades de acoplamiento temporal contra complejidad.

Los Servicios Web son autónomos, o sea, no dependen de sus clientes para funcionar. Esto elimina el acoplamiento de dependencia de los mismos hacia sus clientes. Al eliminar este acoplamiento se simplifica en gran medida la complejidad en cuanto al número y la disponibilidad de los clientes que tiene un servicio.

Contexto Resultante

Beneficios

El beneficio fundamental que se obtiene del uso de la Integración Orientada a Servicios es la interoperabilidad entre sistemas con diferentes arquitecturas técnicas. La interoperabilidad independiza la arquitectura de negocios de la empresa de la tecnología de sus sistemas de información. Al separar ambos conceptos, la empresa adquiere una mayor flexibilidad en cuanto a qué tecnologías usar para implementar sus capacidades de negocio.

Sin sistemas interoperables, el proceso de evolución de los sistemas de información dentro de la empresa se vuelve lento y costoso. El desarrollo está limitado a las capacidades de una tecnología en particular y la empresa tiene que suplir las deficiencias de la misma

desarrollando mecanismos propios. En cambio, logrando interoperabilidad entre los sistemas, se pueden desarrollar módulos que exploten las capacidades particulares de las diferentes tecnologías y después unirlos en un todo coherente. Este enfoque es más ágil y barato a largo plazo. Las empresas que orientan sus arquitecturas de software de esta forma, no sólo están preparadas para la integración de sus sistemas, sino también para integrarse con los sistemas de otra empresa, agilizando el comercio y reduciendo los costos.

Riesgos

La desventaja de los Servicios Web es el precio que se paga en rendimiento al serializar y deserializar[5] los mensajes a XML. Adicionalmente los documentos XML son mucho más extensos que sus equivalentes en un formato binario. Esto incrementa el costo de la transportación de los mensajes.

2.5 Topología de Integración

"La creación de las conexiones a los sistemas y la capa de integración son partes importantes del diseño de la integración. No obstante, es igualmente importante el contexto más amplio que forman estás conexiones y capas a medida que se van uniendo. El diseño del contexto de integración debe especificar los puntos, estructuras y canales que se usan para conectar estos elementos formando un todo coherente. Este contexto recibe el nombre de Topología de Integración." [Trowbridge04]

El análisis de la Topología de Integración gira alrededor de un punto central: el flujo de los mensajes a través de la arquitectura de integración desde la fuente hasta su destino. Como resultado de dicho análisis, en la bibliografía se proponen un conjunto de patrones que van desde la simple Comunicación Punto a Punto, hasta estructuras más complejas como el Broker o el Bus de Mensajes.

La correcta aplicación de los patrones debe apoyarse en la infraestructura física de hardware y software de comunicación, aunque esto no es una condicionante pues las diferentes Topologías de Integración pueden adaptarse a cualquier infraestructura. Un análisis más detallado sobre el tema puede encontrarse en la literatura. [Trowbridge04]

[5] La serialización y deserialización son los nombres que reciben las operaciones de transformación de los datos desde un formato hacia otro y viceversa.

2.5.1 Conexión Punto a Punto

Muchos proyectos de integración parten de la necesidad de conectar dos sistemas y la manera más fácil de hacerlo es usando el patrón de Conexión Punto a Punto. En una Conexión Punto a Punto interviene un solo transmisor y un solo un receptor. Para que pueda efectuarse la comunicación, el transmisor debe conocer la localización del receptor y a menudo también debe traducir el mensaje a un formato que el receptor sea capaz de interpretar.

Al usar el patrón de Conexión Punto a Punto, cada sistema debe determinar las direcciones de cada uno de los nodos con los que necesita comunicarse. Cuando la dirección o los detalles del protocolo de comunicación con un servidor cambian, todos los sistemas que dependen de dicho servidor deben ser actualizados. A medida que la red de aplicaciones crece y la frecuencia de los cambios se incrementa, el costo de la actualización se vuelve significativo.

La mayoría de los escenarios de integración requieren de cierta transformación de los datos entre el sistema fuente y el destino. Adicionalmente, puede requerirse de determinada lógica para establecer condiciones sobre el enrutamiento de los mensajes. En el enfoque punto a punto, el código asociado con ambas tareas esta duplicado en cada nodo. El código duplicado es costoso de escribir, mantener, extender, probar y administrar.

El punto fuerte del patrón Conexión Punto a Punto es su simpleza. Su efectividad para la integración de unas pocas aplicaciones, hace que sea la primera topología que utilizan los desarrolladores. Sin embargo en escenarios más complejos, donde interactúan un mayor número de aplicaciones, es preciso considerar otros patrones.

2.5.2 Broker

A medida que la cantidad de aplicaciones participantes en la solución de integración crece, se hace necesario desacoplar las fuentes de los mensajes de los destinatarios y proveer un control centralizado sobre el flujo de dichos mensajes. Este es precisamente el objetivo del patrón Broker [Hohpe03][Trowbridge04].

El patrón Broker y sus diversos refinamientos también trabajan sobre un conjunto de aspectos básicos de la comunicación:

- *Enrutamiento:* El enrutamiento es el proceso de determinar la localización del sistema objetivo para una comunicación directa o indirecta.
- *Registro de los extremos:* El registro de los extremos es el mecanismo utilizado por un sistema para registrarse en el Broker con el objetivo de que otros puedan descubrirlo.
- *Transformación:* La transformación es el proceso mediante el cuál los datos son convertidos de un formato a otro.

Las características del Broker y de sus diversos refinamientos se resumen en la Figura 2.9

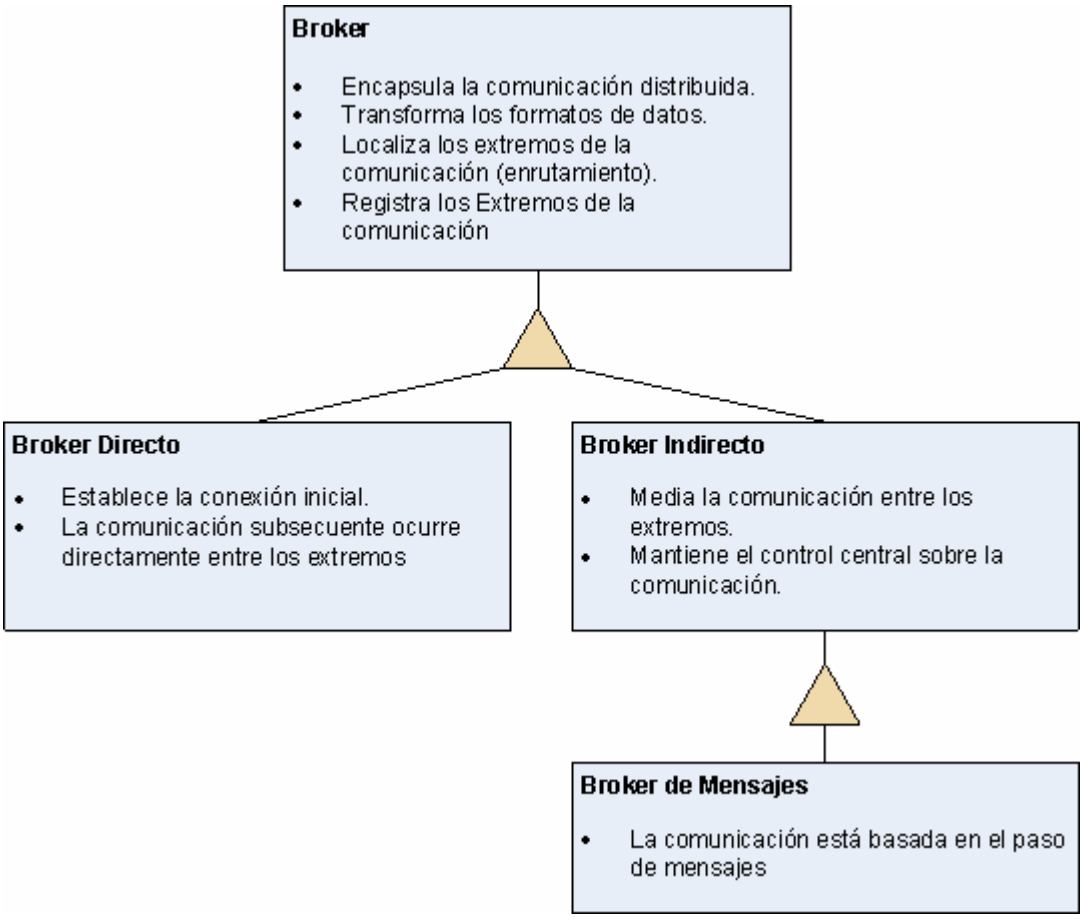


Figura 2.9 Características del patrón Broker y sus refinamientos

El Broker Directo y el Broker Indirecto son especializaciones del patrón general Broker. El Broker Directo solo se encarga del establecimiento inicial de la conexión entre dos sistemas que posteriormente se comunicarán directamente, mientras que el Broker Indirecto controla

toda la comunicación actuando como un intermediario en todo momento. De este último se deriva un patrón más especializado cuando la comunicación está basada en el paso de mensajes llamado Broker de Mensajes.

El Broker Directo posee un mejor rendimiento y soporta una mayor carga, pues solo actúa al inicio de la comunicación. El Broker Indirecto es más útil cuando se requiere un mayor control de la comunicación, por ejemplo, cuando se necesita transformar los datos entre un extremo y otro. Su especialización, el Broker de Mensajes, es uno de los patrones más importantes y usados en los escenarios actuales de integración, tanto es así, que a menudo la bibliografía lo referencia como Broker obviando al resto de la familia.

2.5.3 Bus de Mensajes

El Bus de Mensajes [Hohpe03][Trowbridge04] es una idea que tiene su origen en la electrónica. Consiste en el establecimiento de un medio común para la comunicación entre todas tas aplicaciones. El Bus de Mensajes en la computación parte de tres premisas:

- Un conjunto común de esquemas de mensajes.
- Un conjunto común de mensajes de comandos.
- Una infraestructura compartida para el envío de los mensajes.

Cuando una aplicación envía un mensaje, no necesita hacer una conexión a cada uno de los destinatarios, de hecho, ni siquiera tiene que conocer cuáles son estos destinatarios. Solamente deposita el mensaje en el Bus de Mensajes y este se encarga de hacerlo disponible para todas las aplicaciones que están conectadas al mismo. Las aplicaciones interesadas solo tienen que tomar el mensaje que proviene del Bus de Mensajes.

La principal ventaja de utilizar un Bus de Mensajes es que se logra desacoplar la fuente de los mensajes de los destinatarios, lo que reduce drásticamente la complejidad de la red de interconexiones.

Su principal desventaja radica en que el Bus de Mensajes no distingue entre las necesidades de una aplicación particular. La aplicación sencillamente recibe todos los mensajes que entran al bus. Es su responsabilidad seleccionar los mensajes que le son de interés y desechar el resto. Esto implica un trabajo adicional, por ello, en subsecuentes refinamientos

del patrón, como son el Bus de Mensajes con Publicación – Suscripción basada en Difusión, el Bus de Mensajes Publicación Suscripción Basada en Listas y el Bus de Mensajes con Publicación – Suscripción basada en Contenido, se dan diferentes soluciones a este problema [Trowbridge04].

2.6 Conclusiones Parciales

- El cluster de patrones de integración hace un estudio sistémico de la problemática de la integración y propone un conjunto de soluciones que pueden ser utilizadas por los desarrolladores para el diseño de sus arquitecturas.
- Haciendo una analogía con NGOSS; con la excepción de SID, todos los paradigmas de TNA con respecto a la integración tienen su equivalente en patrones del cluster de patrones de integración.

CAPÍTULO 3. TECNOLOGÍAS DE INTEGRACIÓN

El desarrollo de soluciones de integración constituye una de las fuentes de ingresos de la industria de la computación. La producción de tecnologías de integración es una línea que ha ido ganando terreno dentro de los principales productores como Microsoft, Bea, Oracle e IBM entre otros. Cada año aparecen nuevos productos destinados a solucionar problemas de integración o a facilitar el desarrollo de aplicaciones distribuidas. Cada año los vendedores enuncian las ventajas de sus productos y la forma en que resuelven este u otro escenario.

Desde el punto de vista de los desarrolladores, no es tarea fácil decidir cuáles tecnologías utilizar en un mercado tan cambiante. Toma tiempo determinar hasta dónde llegan las capacidades reales de las tecnologías en un medio donde abunda la propaganda comercial. Un análisis pobre de las capacidades de las tecnologías puede conducir a implementaciones que no satisfagan los requerimientos operacionales de la solución.

Si los desarrolladores deciden basar su diseño en patrones, ¿no podrían los patrones ayudar a hacer una selección de las tecnologías más idóneas para su implementación? Los escritores de patrones a menudo proponen tecnologías como parte complementaria al patrón. Algunos optan por hacer un refinamiento del patrón general dirigido a la implementación del mismo con una tecnología específica. Esto ya constituye una buena recomendación, sin embargo, no todas las tecnologías tienen la suerte de ser referenciadas por la literatura de patrones.

Más allá de la simple recomendación de una solución o tecnología, cada patrón hace un detallado análisis de las fuerzas que intervienen en el problema. Este análisis sirve como referencia para que el equipo de desarrollo evalúe las capacidades de determinada tecnología para implementar el patrón. Además de ello, el patrón hace una descripción del contexto resultante en la cual expone los riesgos que se corren al aplicarlo. La forma en que la tecnología es capaz de enfrentar estos riesgos constituye otra referencia para su evaluación.

En el presente capítulo se hace un análisis de un conjunto de nueve tecnologías de integración que responden fundamentalmente a la línea Microsoft. La causa de esta selección es el predominio del sistema operativo Windows y otras tecnologías de Microsoft a lo largo de la infraestructura de software de ETECSA.

El análisis de las tecnologías toma como referencia los Patrones de Integración para hacer la evaluación como se explicó anteriormente. Los criterios expuestos en dicha evaluación determinaron la selección de las tecnologías propuestas para la Arquitectura de Integración de ETECSA.

3.1 Microsoft SQL Server Data Transformation Services

Microsoft SQL Server Data Transformation Services (DTS) es una tecnología de propósito general para la extracción, trasformación e introducción de los datos desde unas fuentes de datos hacia otras. Como su nombre lo indica, forma parte de las tecnologías adjuntas al gestor de datos Microsoft SQL Server a partir de su versión del 2000.

La arquitectura de DTS está diseñada para conectarse directamente a las fuentes de datos de las aplicaciones. Esto lo logra a través de un conjunto de proveedores de datos que le permiten acceder a cualquier Base de Datos relacional y otras fuentes como ficheros de Excel, ficheros de texto, etc. Es su flexibilidad de conexión a los datos crudos la razón fundamental que la promueve como una tecnología a considerar en cualquier escenario de Integración de Datos.

Mediante el uso de una herramienta gráfica (DTS Designer) los desarrolladores pueden diseñar procesos de sincronización que incluyan operaciones de extracción, trasformación y consolidación de los datos. Cada proceso constituye un paquetes de transformación (DTS Package) que es ejecutado por el gestor de transformaciones (DTS Manager).

Las ventajas de utilizar DTS para las operaciones con datos son múltiples y muy variadas, entre ellas se pueden mencionar las siguientes:

- La arquitectura de DTS encapsula muchos de los detalles específicos de cada una de las fuentes de datos y permite a los desarrolladores concentrarse en la lógica sincronización.

- Los procesos de sincronización en DTS, a diferencia de otras tecnologías, incluyen la transformación de los datos, lo que permite integrar dos modelos de datos diferentes (permite resolver las disonancias semánticas).
- Los paquetes de transformación se pueden definir y modificar dinámicamente lo que agrega flexibilidad a las soluciones.
- La lógica de los procesos en DTS es definida y controlada en una capa independiente disminuyendo la complejidad de las aplicaciones.
- Los procesos se diseñan gráficamente mediante la combinación de componentes prefabricados, código en SQL y el uso de lenguajes *script*. No se requiere de un alto grado de conocimientos tecnológicos ni de una gran cantidad de tiempo para diseñar un proceso.
- Los procesos en DTS permiten el manejo de transacciones, asegurando así la integridad de los datos ante los fallos.

La Integración de Datos, según se explica en el patrón, está sujeta a los cambios de las estructuras internas de los datos. En este sentido, DTS posee varias características que contribuyen a disminuir el impacto que tienen los mismos y con ello aumentar la mantenibilidad de la solución. Entre ellas se pueden destacar las siguientes:

- La capacidad de mantener toda la lógica de integración en una misma Capa de Integración centralizada, permite que el impacto no se propague más allá de dicha capa.
- La capacidad de modificar los paquetes una vez que se han puesto en explotación, permite resolver los problemas sin tener que detener el funcionamiento de toda la solución, basta con detener el paquete afectado.
- La simplificación, mediante una herramienta gráfica, del desarrollo de los paquetes de transformación, permite bajar los costos asociados a las modificaciones necesarias.

La capacidad de DTS para resolver las disonancias semánticas, mediante el uso de múltiples herramientas de transformación y consolidación, la hace una tecnología útil para la implementación de una Capa de Agregación de Entidades.

El desarrollo de DTS como tecnología de integración de datos continúa. Ya se avizora una nueva versión como parte del Microsoft SQL Server 2005 (Yukon). Entre los cambios fundamentales de esta versión está el soporte para la integración con fuentes de datos en XML. Como soporte para los datos, XML tiene un papel central en las nuevas tecnologías de integración.

3.2 Microsoft Message Queuing

Microsoft Message Queuing (MSMQ) es una arquitectura de colas de mensajes integrada al sistema operativo Microsoft Windows, cuya función es brindar soporte para la comunicación entre las aplicaciones. Sus características más relevantes como arquitectura son las siguientes:

- Garantiza una comunicación confiable entre aplicaciones ante las latencias y los fallos de la red, ya que los mensajes enviados son salvados en el disco hasta que se verifica que han llegado a su destino.
- Permite enlistar el envío de mensajes en transacciones distribuidas usando Microsoft Distributed Transaction Coordinator (DTC).
- Permite el establecimiento de distintos niveles de prioridad a los mensajes. Con ello se garantizar la llegada en tiempo de los mensajes de alta prioridad en escenarios con alta congestión.
- Implementa mecanismos de seguridad que incluyen la autenticación y encriptación de los mensajes. Permite administrar los privilegios de los usuarios mediante un mecanismo de control de acceso en las colas.
- Se integra con Active Directory permitiendo registrar las colas como objetos del Directorio de Windows. Esto facilita las tareas de localización y administración de las colas.
- Posee capacidades de publicación – suscripción mediante una arquitectura de reglas y disparadores.
- Permite el uso de HTTP y HTTPS como protocolos de transporte.

La comunicación por MSMQ ocurre fundamentalmente sobre TCP a través de un protocolo propietario, lo que limita su uso a la red privada. Con el objetivo de solucionar esta

limitante, a partir de la versión 3.0 se incluyó el paso de mensajes mediante SRMP (SOAP Reliable Messaging Protocol) sobre HTTP o HTTPS. Usando estos protocolos, se logró establecer una comunicación segura a través de *proxies* y cortafuegos convencionales, con lo que se extendió el uso de MSMQ a toda la Internet.

Vista a través de los patrones de integración, MSMQ posee excelentes características para la implementación de una Capa Intermedia Orientada a Mensajes. Sus altos niveles de fiabilidad la promueven como mecanismo de comunicación a bajo nivel en muchas tecnologías de integración. Ejemplos de ello son el uso MSMQ para la comunicación asincrónica en COM+ y la arquitectura de mensajería de Biztalk.

El uso de MSMQ también permite la implementación de topologías complejas como es el caso del Bus de Mensajes. En este caso, la cola de MSMQ constituye un medio común que puede ser usado para el acceso compartido de varias aplicaciones. Esto se puede combinar con las capacidades de publicación – suscripción de MSMQ para lograr variantes más refinadas del patrón como el Bus de Mensajes con Publicación – Suscripción Basada en Listas y el Bus de Mensajes con Publicación – Suscripción basada en Contenido.

3.3 COM+

COM+ es el resultado evolutivo de Microsoft Common Object Model (COM) y Microsoft Transaction Server (MTS). Como tecnología de componentes, brinda un conjunto de servicios que incrementan las potencialidades de los mismos. Entre estos servicios, tres son destacables en cuestiones de integración: el primero permite comunicación remota con los componentes vía Microsoft Distributed Common Object Model (DCOM), el segundo permite dicha comunicación a través de Microsoft Message Queue (MSMQ) y el tercero permite la comunicación por SOAP.

Desde el punto de vista de la integración, la principal restricción de las versiones iniciales de COM+ fue su incapacidad de interoperar con otras tecnologías. Al utilizar un protocolo propio para la comunicación entre los objetos, COM+ sólo permitía interoperar con aplicaciones que implementaban DCOM. Esto es bastante restrictivo pues DCOM es una

tecnología exclusiva de las plataformas Microsoft Windows. Además, RPC[6], protocolo usado por DCOM para la comunicación, es muy sensible a ataques por lo que la mayoría de las empresas prefieren restringir su uso solo a pequeños segmentos de su red privada.

Al utilizar MSMQ, COM+ adquirió las ventajas del trabajo con una arquitectura de colas para la comunicación con sus componentes. El uso de MSMQ permitió ganar en fiabilidad y seguridad a las aplicaciones. Sin embargo, al igual que DCOM, MSMQ es una tecnología exclusiva de la plataforma Windows.

Con el objetivo de resolver las limitantes de la plataforma problemas y exponer los componentes de COM+ para una integración abierta a Internet, a partir de su versión 1.5 COM+ implementa la comunicación mediante SOAP, permitiendo exponer los componentes como Servicios Web. Con ello, COM+ gana un conjunto de ventajas en la interoperabilidad y seguridad similares a los ASP.Net Web Services[7].

Si se utiliza COM+ para implementar la capa de negocios de una aplicación, la Integración Funcional de la misma queda garantizada; mediante una forma de comunicación u otra, dichos componentes son accesibles de manera remota. Dentro de la Integración Funcional, COM+ se ajusta al patrón de Integración Mediante Objetos Distribuidos.

Además de sus posibilidades para la integración, COM+ posee una gran cantidad de ventajas como tecnología de componentes, por lo que es muy recomendable su uso para la implementación de una Capa de Negocio que permita la Integración Mediante Objetos Distribuidos.

3.4 CORBA

Las siglas CORBA provienen de Common Object Request Broker Arquitecture y dan nombre a un conjunto de especificaciones del Object Management Group (OMG) para solucionar el problema de la interoperabilidad en la programación de componentes.

[6] Remote Procedure Call (RPC)
[7] ASP.Net Web Services es una tecnología para construir Servicios Web. Sus ventajas y desventajas serán analizadas en posteriores epígrafes.

CORBA utiliza la idea de establecer un lenguaje intermedio de representación de interfaces independiente de la tecnología (CORBA IDL), para la descripción de los metadatos necesarios en la comunicación entre dos componentes. Este lenguaje, al ser compilado, genera dos objetos intermediarios que conforman ambos extremos de un canal de comunicación. Los componentes en cada extremo se comunican con los intermediarios mediante el uso de un conjunto de reglas de mapeo, las cuales están también especificadas por OMG.

Este enfoque exime a los fabricantes de lenguajes, de la implementación de protocolos de comunicación, su única responsabilidad es la conversión de los datos desde su lenguaje específico hacia IDL y viceversa. Esta separación de las responsabilidades permite que los protocolos de comunicación evolucionen independientemente de los componentes que hacen uso de los mismos.

Desde el punto de vista de la Integración Funcional y más específicamente de la Integración mediante Objetos Distribuidos, CORBA presenta una especificación que resuelve los problemas de interoperabilidad entre plataformas y tecnologías heterogéneas, al mismo tiempo que mantiene la ilusión de simpleza ante los desarrolladores.

Es destacable en la arquitectura de CORBA, la existencia del Interoperable Naming Service (INS) que actúa como un Broker Indirecto agregando más flexibilidad en la topología de integración y preparando las aplicaciones para la escalabilidad.

CORBA también incluye servicios de comunicación asincrónica y de manejo de colas que permiten lidiar con la latencia de la redes y los fallos en la comunicación. Además, maneja el concepto de **calidad de servicio**, cosa que otras tecnologías no hacen.

Desde el punto de vista de la seguridad de los protocolos, la OMG ha elaborado especificaciones para el desarrollo de cortafuegos y proxies para IIOP soportado sobre SOCKS, SSL, TCP, etc.

Desafortunadamente, CORBA no ha sido asimilado con la misma fuerza por todos los productores de tecnologías , hecho que era vital para su objetivo de interoperar. Mas allá de los intereses de los vendedores, está el hecho de que cada productor necesita desarrollar un

ORB (Object Request Broker) que se encargue de la comunicación entre CORBA y una tecnología específica, lo cual no es un proceso sencillo. A pesar de ello, el uso de CORBA para la Integración mediante Objetos Distribuidos es muy popular en algunos entornos, por ejemplo en Java.

3.5 .Net Remoting

El nombre de .Net Remoting responde a una tecnología nativa del Microsoft .Net Framework para la comunicación entre objetos, los cuales pueden estar en diferentes procesos de una misma computadora o en computadoras separadas. La comunicación en .Net Remoting ocurre de manera transparente para los desarrolladores, lo que facilita la concepción de soluciones distribuidas.

Entre las facilidades destacables de .Net Remoting está la dualidad de protocolos de transporte. El primero de ellos, es un protocolo binario de comunicación, accesible por TCP, para los escenarios de altos requerimientos de rendimiento y bajos requerimientos de seguridad. El segundo, es un protocolo basado en SOAP por HTTP para los escenarios de Internet. Es de destacar, que el protocolo de HTTP no requiere de un servidor Web, lo que aligera las aplicaciones basadas en .Net Remoting.

Gracias al uso de SOAP por HTTP, .Net Remoting puede utilizarse en escenarios con elevados requerimientos de seguridad. Además, el uso de HTTP como protocolo de transporte, permite la utilización de proxies convencionales de HTTP para el acceso desde y hacia la red interna. Otro aspecto importante para la seguridad es que SOAP, al ser un protocolo basado en XML que a su vez es texto plano, permite el uso de cortafuegos y Sistemas de Detección de Intrusos (DSI) profesionales. Todo ello eleva la confianza de los administradores en la tecnología.

Entre las desventajas de .Net Remoting está su poca interoperabilidad. Su uso está limitado a las aplicaciones soportadas sobre .Net Framework, lo cual tiene dos implicaciones directas. La primera es que aunque teóricamente el .Net Framework no está atado a ningún lenguaje de programación (de hecho ya existen múltiples lenguajes como C#, Visual Basic .Net, Java#, Delphi .Net que generan código para .Net), la inmensa mayoría de las aplicaciones legadas no hacen uso de .Net Framework. La segunda es que aunque

teóricamente el .Net Framework, al igual que la Máquina Virtual de Java, permite el desarrollo de aplicaciones multiplataformas, todo parece indicar que por el momento el soporte para .Net Framework será un privilegio exclusivo de Microsoft Windows.

Con respecto al soporte de carga, .Net Remoting no está preparado para lidiar con grades cantidades de usuarios concurrentes, por lo tanto la solución debe considerar este aspecto.

El uso de .Net Remoting permite la comunicación con los objetos de la Capa de Negocio siguiendo el patrón de Integración mediante Objetos Distribuidos. La aplicación de esta tecnología es recomendable si se puede garantizar que todos los clientes estén soportados sobre .Net Framework y la carga total de la aplicación servidor sea controlable.

3.6 ASP.Net Web Services

Otra de las tecnologías nativas del Framework de .Net es ASP.Net Web Services, la cual permite la comunicación entre aplicaciones mediante la exposición de las funcionalidades de la capa de negocio en forma de Servicios Web. En cuestiones de interoperabilidad, los Servicios Web son la variante más aceptada y es por ello que ASP.Net Web Services es un candidato a considerar al enfrentarse a la integración de aplicaciones tecnológicamente heterogéneas.

Entre los beneficios de la tecnología se encuentra su sencillez. Desde el punto de vista del programador la exposición y comunicación con los Servicios son procedimientos meramente declarativos. La tecnología encapsula todos los detalles de la comunicación, a la vez que provee mecanismos que permiten personalizar la misma en caso necesario. Esto permite a los desarrolladores obtener rápidamente una primera versión y posteriormente personalizarla según los requerimientos.

Los ASP.Net Web Services utilizan SOAP sobre HTTP y HTTPS como protocolo de comunicación, lo que les permite satisfacer altos requerimientos de seguridad y accesibilidad mediante su combinación con *proxies* y cortafuegos profesionales.

Una cuestión a tener en cuenta al usar ASP.Net Web Services es que por defecto dicha tecnología genera los contratos automáticamente. Se ha constatado en la práctica que a medida que aumenta la complejidad de las interfaces, estos contratos automáticos se

vuelven carentes de semántica y en algunos casos de información estructural. Esto provoca problemas de interoperabilidad al no ofrecerse la información suficiente para la comunicación. Dichos problemas puede ser evitados diseñando los contratos manualmente y generando, a partir de los mismos, el Servicio Web. El Contract First, nombre que recibe está técnica, aunque permite la concepción de mejores contratos, tiene la desventaja de que requiere de un nivel de conocimientos sobre XMLSchemas y WSDL que algunos desarrollares no poseen.

3.7 Microsoft Web Services Enhancements

Microsoft Web Services Enhancements (WSE) es una tecnología no nativa del Framework de .Net enfocada a elevar las posibilidades operacionales de los servicios.

Entre las posibilidades operacionales de los Servicios Web de WSE está la seguridad. La misma está implementada mediante el paso de credenciales en la cabecera del mensaje de acuerdo con la especificación estándar WS-Security. Este mecanismo permite un manejo de la seguridad libre de plataforma y de protocolo de comunicación. La seguridad en WSE incluye la autenticación, la encriptación de los mensajes y el manejo de los perfiles de seguridad basándose en el conjunto de especificaciones estándares WS-Trust, WS-SecureConversation y WS-Security Profile respectivamente.

Otra característica sobresaliente de WSE es que permite la comunicación con los Servicios Web por múltiples protocolos de transporte. Esto significa que un mismo servicio puede comunicarse a través de HTTP o TCP indistintamente. En el caso de HTTP, WSE utiliza la arquitectura de ASP.Net como soporte, sin embargo, en TCP no se requiere de un servidor Web para la comunicación. La dualidad de protocolos permite a los desarrolladores elegir cuál resulta más apropiado según sus requerimientos de carga y disponibilidad de hardware. A esto se le agrega que el modelo de transporte de WSE no está limitado solamente a estos dos protocolos por defecto, sino que puede ser extendido para adicionarle nuevos protocolos. Por ejemplo, está disponible el transporte de MSMQ para WSE que permite a los servicios aprovechar las capacidades de fiabilidad de esta arquitectura de mensajería.

WSE maneja el concepto de SOAP con adjuntos. Los adjuntos son fragmentos de información potencialmente grandes que viajan con el mensaje. Un adjuntos puede ser

cualquier información, desde una imagen hasta un fichero texto. Estos contenidos son incrustados al mensaje SOAP mediante el protocolo DIME (Direct Internet Message Encapsulation), con lo que se elimina la necesidad de serialización y deserialización. El manejo de adjuntos mejora los indicadores de rendimiento de los servicios Web, fundamentalmente ante contenidos largos y difíciles de convertir a XML. Con ello se soluciona la principal deficiencia de la integración mediante Servicios Web.

WSE es una tecnología que permite la Integración Orientada a Servicios en escenarios con altos requerimientos de flexibilidad, interoperabilidad y seguridad. Sin embargo, este no es el único uso de WSE en la integración.

WSE presenta una arquitectura de filtros SOAP para el procesamiento de los mensajes. Dichos filtros se acoplan en una especie de línea de procesamiento que va procesando las diferentes informaciones de la cabecera del mensaje. Esta arquitectura permite la implementación de otros conceptos como en enrutamiento y la trasformación de los mensajes. El enrutamiento de mensajes es un mecanismo nativo de la tecnología y la transformación puede ser implementada con facilidad, esto hace que WSE sea una arquitectura factible para la implementación de topologías de integración más complejas, como el Broker de Mensajes.

3.8 Indigo

Con el transcurso de los años se ha ido acumulando experiencia sobre el desarrollo de sistemas conectados. Basada en esta experiencia, Microsoft se ha enfrascado en la construcción de una nueva tecnología que ayude a los programadores a enfrentar los desafíos de la integración de sistemas. Esta tecnología, recibe el nombre de Indigo.

Indigo es una plataforma orientada a simplificar el proceso de desarrollo de sistemas conectados. Según la visión de Microsoft, la verdadera explosión de las capacidades intrínsecas de la integración de sistemas está condicionada al desarrollo de tecnologías que simplifiquen y abaraten la construcción de dichas arquitecturas [Box03]. Es por ello que Indigo no solo está formada por un revolucionario modelo de objetos que encapsulan los conceptos más modernos del desarrollo de Servicios Web, sino que además complementa la plataforma con un conjunto de herramientas orientadas a facilitar el trabajo.

Uno de los factores que dificulta el desarrollo de aplicaciones conectadas en la actualidad, es que cada una de las tecnologías existentes brinda un subconjunto de facilidades propias. Esto hace que los desarrolladores tengan que conocer diversas tecnologías y combinarlas de ingeniosas maneras para lograr satisfacer los requerimientos de los problemas. Indigo pretende solucionar esta situación reuniendo bajo un único modelo homogéneo las ventajas de todas las tecnologías precedentes de Microsoft, o sea, los Servicios Web de Indigo estarán capacitados para ser altamente interoperables, soportar transacciones distribuidas, comunicarse de forma confiable, manejar políticas de seguridad y mantener estado; además serán independientes del protocolo transporte y podrán aprovechar las facilidades de las arquitecturas de colas. Todo ello permitirá a los programadores contar con un gran arsenal de tecnologías para resolver los más diversos problemas sin tener que abandonar el contexto de Indigo.

Otro problema que pretende solucionar Indigo es qué hacer con el software implementado sobre tecnologías anteriores. Para ello, Microsoft está trabajando en tres líneas fundamentales: la primera consiste en hacer que Indigo esté disponible a una amplia gama de lenguajes de programación (C#, Visual Basic, C++, etc); la segunda consiste en que las aplicaciones basadas en tecnologías más antiguas como ASP.Net Web Services, COM+ y MSMQ sean capaces de comunicarse con los nuevos servicios de Indigo y finalmente, en caso de que programadores decidan mejorar las potencialidades de sus aplicaciones migrándolas hacia Indigo, la tercera línea trabaja en un conjunto de herramientas y guías para facilitar este proceso.

Uno de los conceptos más modernos que introduce Indigo es el soporte a nivel tecnológico para la evolución de las aplicaciones. Se plantea que la evolución de las aplicaciones debe ir a la par con la evolución de los negocios, esto es tema de un intenso debate internacional sobre cómo construir sistemas más mantenibles que puedan evolucionar con la velocidad requerida a un costo apropiado. Indigo introduce en su infraestructura tecnológica un conjunto de prácticas que aumentan la flexibilidad de las aplicaciones para ajustarse a escenarios cambiantes.

Desde el punto de vista de la Integración Orientada a Servicios, Indigo se propone llevar el concepto de servicio a un nivel mayor de soporte tecnológico. Esto seguramente tendrá implicaciones en la visión que ahora se tiene de la integración de sistemas. Los conceptos de patrones que se manejan en Indigo ponen al desarrollo de software, un paso más cerca de lo que algunos vaticinan como el futuro de los sistemas conectados: las Aplicaciones Orientadas a Servicios.

Aunque Indigo todavía está en proceso de desarrollo, y de hecho su primera versión está planificada para el año 2006, el conjunto de mejoras que se avizoran a través de las entrevistas con los desarrolladores y la vista previa de la tecnología, ha captado la atención de la comunidad de desarrollo de aplicaciones de integración. Por tanto, es una tecnología a tener en cuenta en la proyección futura de las infraestructuras de software del presente.

3.9 Microsoft Biztalk Server

Microsoft Biztalk Server es uno de los servidores empresariales de Microsoft totalmente orientado a la integración de aplicaciones. Está orientado a los problemas de integración de procesos, pero su versátil diseño le permite resolver otros escenarios de integración.

Cuando se piensa en Biztalk desde el punto de vista de la integración, a menudo se suele sobrestimar sus capacidades de solución. De allí que exista una cierta tendencia al sobreuso de esta tecnología. Conciente de dicha tendencia, Trowbridge [Trowbridge04] propone una serie de refinamientos que muestran cómo implementar diversos patrones con Biztalk.

El corazón de Biztalk es la mezcla de dos arquitecturas: la arquitectura de mensajería y el gestor de procesos, cada una de las cuales juega un importante rol en su funcionamiento.

Una de las funciones de Biztalk es actuar como un intermediario facilitando la comunicación entre aplicaciones, por ello, su arquitectura de mensajería es una pieza fundamental.

Biztalk abstrae todo el mecanismo de comunicación al paso de mensajes XML entre aplicaciones. Esta abstracción es uno de los factores que más influye en la versatilidad de Biztalk. Como se analiza por Hohpe [Hohpe03], la comunicación mediante el paso de

mensajes es la base para la solución de los diferentes retos de la comunicación a través de un conjunto de patrones.

Otro aspecto relevante de Biztalk es su capacidad para obtener datos a través de diferentes mecanismos de comunicación. Biztalk se comunica con el exterior a través de unos componentes llamadas **adaptadores**. Cada clase de adaptador implementa un protocolo diferente de comunicación. Mediante la configuración de las instancias de los adaptadores, Biztalk puede recibir y enviar datos a un fichero texto, una base de datos, un componente de COM+ y un Servicio Web entre otros. Si lo anterior no fuese suficiente, aparte de los adaptadores de propósito general que acompañan a Biztalk, existen otra gran cantidad disponibles en Internet. Además, la arquitectura de adaptadores es extensible y permite el desarrollo de nuevos adaptadores de propósito específico. Mediante los adaptadores, Biztalk se puede integrar a los tres niveles de la aplicación: datos, función y presentación.

Otros componentes fundamentales dentro de la mensajería de Biztalk son las **tuberías**. Una tubería es una línea de procesamiento donde cooperan diversos componentes, cuya labor fundamental es la transformación de los mensajes desde un formato determinado hacia XML y viceversa. La conversión de los mensajes hacia XML facilita la comunicación entre partes heterogéneas, esto es otra de las claves de Biztalk. Además de su función principal, las tuberías ejecutan otras tareas como la autenticación y la encriptación de los mensajes aumentando la seguridad de la comunicación. Al igual que los adaptadores la arquitectura de tuberías es extensible.

Los adaptadores y las tuberías se correlacionan en los **puertos**. Cada puerto representa un punto lógico de comunicación con Biztalk. Mediante el establecimiento de puntos lógicos de comunicación independientes de protocolo y formato, Biztalk logra reducir drásticamente el impacto que producen los cambios de protocolo y formato en la comunicación. Este aspecto es muy importante cuando se considera la creación de una arquitectura de comunicación que evolucione a un bajo costo.

Una vez que los mensajes están en XML según un esquema específico, son colocados en el **buzón de mensajes**. Este buzón es un Bus de Mensajes hacia donde tributan los puertos de recepción. Los puertos de envío pueden estar suscritos directamente al buzón mediante un

mecanismos de reglas llamadas **filtros**. Todo ello conforma una arquitectura de publicación – suscripción que completa el patrón Bus de Mensajes.

Estas características resumen la arquitectura de mensajería de Biztalk, la cual por si misma permite la solución de diferentes escenarios de integración, como por ejemplo la implementación de un Broker de Mensajes como señala Trowbridge [Trowbridge04]. Este broker combina las capacidades de enrutamiento y transformación con una arquitectura de publicación – suscripción basada en contenido. Si a ello se le añade las posibilidades de Biztalk para la comunicación con diversos protocolos y en múltiples formatos de datos y sus capacidades intrínsecas de autenticación y encriptación, resulta que el Broker de Mensajes implementado con Biztalk posee amplias posibilidades operacionales y satisface un conjunto mucho más amplio de requerimientos.

La otra parte de Biztalk tiene que ver con su función fundamental, la implementación de procesos de negocio según el patrón de Integración de Procesos. Tanto es así, que la literatura propone La Implementación de Integración de Procesos con Biztalk 2004 como un refinamiento de dicho patrón [Trowbridge04].

Los procesos de negocio en Biztalk son definidos mediante un lenguaje de definición de procesos llamado XLANG. Un proceso de negocio definido en dicho lenguaje recibe el nombre de **orquestación** y contiene la lógica necesaria para que Biztalk coordine la cooperación entre varias aplicaciones.

El lenguaje XLANG incluye operaciones para trabajar con conceptos intrínsecos de la integración de procesos como son: la comunicación asincrónica, la correlación de los mensajes, el soporte para transacciones de larga duración y una infraestructura para la detección, manipulación y acción ante situaciones de error. Estos conceptos, que son difíciles de implementar usando lenguajes de programación tradicionales, están implementados por defecto en Biztalk y son definidos mediante una herramienta gráfica. Todo ello contribuye a que los procesos de negocio sean más fáciles de definir, implementar y cambiar, lo que tiene una influencia directa en el costo de la solución de integración.

Para la comunicación con las aplicaciones participantes en un proceso de negocio, Biztalk se apoya en su arquitectura de mensajería. Este enlace ocurre a través de los puertos, quedando separada la lógica del proceso de negocio de los detalles de la comunicación con las aplicaciones. Esta concepción es sumamente flexible ante el impacto de los cambios en el proceso de negocio y en las aplicaciones subyacentes.

El gestor de procesos de Biztalk posee otras dos características que merecen su mención. La primera es la capacidad de exponer sus orquestaciones como servicios, con la cual ofrece sus facilidades de gestor de procesos para una integración a más alto nivel. La segunda es la portabilidad de sus orquestaciones hacia BPEL, un lenguaje estándar de definición de procesos.

Aparte de su funcionalidad primaria, Biztalk también posee otros dones que lo hacen interesante desde el punto de vista de los negocios. Uno de ellos es el módulo de monitoreo de la actividad del negocio (BAM), que permite darle seguimiento a los procesos de negocio, encontrar los cuellos de botella y obtener un conjunto de estadísticas sobre el funcionamiento de las actividades en la empresa. Mediante el BAM, la empresa se retroalimenta con informaciones que la pueden ayudar a mejorar y optimizar sus mecanismos de funcionamiento.

Otro aspecto importante es el rol que juega Biztalk en el comercio B2B. Biztalk permite la participación de socios comerciales dentro de los procesos de negocio de la empresa. Con ello agiliza la actividad comercial y aumenta la competitividad de la empresa.

Es de destacar que aunque a simple vista el comercio electrónico no tiene una relación directa con la integración de aplicaciones, gran parte de los expertos en integración trabajan en estas áreas. Esto no es casual, pues la integración intra – empresarial de aplicaciones casi siempre antecede a la integración inter – empresarial. En este último nivel es donde los retos de integración son mayores y es por ello que la empresa debe proyectar sus soluciones de integración para un futuro de comercio B2B.

Como se puede apreciar Biztalk es una verdadera “navaja suiza”. Sin embargo, existen algunos aspectos a considerar antes de emplear dicha tecnología. El costo de Biztalk es alto

dado que es un producto de avanzada de Microsoft. Al costo de la licencia se suman los altos requerimientos hardware y software. Es por ello que antes de usar Biztalk, es preciso analizar la dimensión del problema que se intenta solucionar. Por último, Biztalk es un punto central en la comunicación, por tanto un cuello de botella en Biztalk representa un cuello de botella en toda la empresa. En este caso deben considerarse las opciones que brinda Biztalk para la distribución de la carga.

3.10 Conclusiones Parciales

- Los patrones de integración permiten hacer una evaluación de capacidad de las tecnologías para resolver las fuerzas en un escenario de integración y para enfrentar los riesgos de las soluciones propuestas.
- Las tecnologías analizadas son suficientes para una rápida implementación de la mayoría de los patrones de integración en un entorno Windows.

CAPÍTULO 4. ARQUITECTURA DE INTEGRACIÓN

El estudio de las tendencias actuales de la gestión, ha provocado que la empresa de telecomunicaciones ETECSA abrace cada vez con más fuerza el paradigma de la gestión orientada a procesos. Sin embargo, al no contar con la infraestructura de software necesaria para ello, la necesidad de construir una plataforma de gestión ya es un hecho palpable. La construcción de dicha plataforma es particularmente compleja y a la vez inaplazable.

En un futuro la Plataforma de Gestión de ETECSA estará compuesta por múltiples sistemas que apoyen las diferentes operaciones en la gestión. Estas aplicaciones deberán cooperar para llevar a cabo los diferentes procesos de la empresa y por tanto este es un escenario propicio para la aplicación de todo tipo de soluciones de integración.

Todas las bondades de la integración de aplicaciones pueden cobrar un alto precio si esta no se planifica cuidadosamente. Sin el establecimiento de políticas bien definidas, toda la plataforma pudiera derivar en un amasijo de aplicaciones interconectadas imposible de mantener y modificar. En este punto, la empresa se vería obligada a desechar una enorme cantidad de trabajo para llevar la plataforma a un estado en que se pudiera evolucionar.

Por otro lado, integrar aplicaciones que no hayan sido concebidas al efecto puede ser sumamente difícil y en ocasiones no factible. Todo eso pudiera evitarse si las empresas, al concebir o comprar las nuevas aplicaciones, tomaran en cuenta la forma en que estas se integran dentro de la plataforma.

En el proceso de desarrollo de la Plataforma de Gestión gran parte del esfuerzo estará dedicado a la planificación y elaboración de las soluciones de integración, esfuerzo que a la larga significa gasto de tiempo y recursos. Es interés de los directivos conocer si existe alguna forma de prepararse para las futuras soluciones de integración, de manera que puedan tener una rápida implementación en el menor tiempo posible. La solución en este caso es la construcción de una Arquitectura de Integración que brinde soporte a las soluciones de integración que se avecinan.

En este capítulo se muestra el diseño teórico de la Arquitectura de Integración, así como un conjunto de consideraciones que sirven de base para la construcción de una plataforma de

gestión orientada a los servicios para la red de ETECSA. También se incluye una propuesta de las tecnologías necesarias para su implementación.

4.1 Plataforma de Gestión de ETECSA

En el marco de los esfuerzos que hace la empresa por construir su plataforma de gestión se encuentra el proyecto para la automatización de los procesos de Operación y de Mantenimiento. El proceso de Operación tiene por finalidad la provisión y/o modificación de los servicios, de acuerdo a las necesidades y requerimientos del cliente externo (usuario de la red). La salida principal del proceso es la provisión, modificación o suspensión del servicio al cliente. Por su parte, el proceso de Mantenimiento tiene por objetivo identificar y resolver problemas en el servicio, tanto ante alarmas automáticas como ante indicaciones de los clientes internos (responsables de la supervisión de la red, de la calidad del servicio, de la provisión de nuevos servicios, etc.) o externos (reclamaciones de los usuarios), así como a partir de actividades de control cotidianas y planificadas; con el fin de asegurar la continuidad del servicio, o bien efectuar su reactivación mediante oportunas intervenciones correctivas.

Los primeros pasos en dicha labor arrojaron la necesidad de construir un conjunto inicial de sistemas de apoyo a la gestión. Ellos son:

- *El Inventario de la Red:* que maneja la información de la red, tanto física como lógica y los servicios que están soportados sobre esta.
- *El Gestor de Alarma:* encargado de la detección y seguimiento de los fallos en la red.
- *El Gestor de la Fuerza de Trabajo:* encargado de la gestión de los recursos humanos desde el punto de vista técnico.
- *El Sistema de Provisión:* encargado de manejar los acuerdos de servicios y de monitorear el proceso desde la solicitud de un servicio hasta la instalación del mismo.

A la construcción de estos sistemas se le agrega la implementación de los procesos de gestión de Operación y de Mantenimiento antes mencionados.

4.2 Problema de la Fragmentación de la Red

En los últimos años ETECSA ha hecho grandes inversiones con el objetivo de mantener la red cubana de telecomunicaciones a la altura de las necesidades y perspectivas del país. La compra del equipamiento ha sido efectuada a varios proveedores y en diferentes momentos. Todo ello ha derivado en una red donde se combinan equipos y sistemas de gestión de múltiples tecnologías y fabricantes.

Cada fabricante oferta sistemas que permiten gestionar el equipamiento de su manufactura. Entre ellos se pueden diferenciar tres clases de sistemas: los Sistemas de Gestión de los Elementos, que permiten gestionar un equipo específico, los Sistemas de Gestión de la Red, que gestionan múltiples equipos y sus conexiones como una red y los Sistemas de Gestión de los Servicios que manejan los servicios soportados sobre redes de equipos. Estos últimos tienen la limitante de no ofrecer un soporte completo a la gestión de los servicios y solo gestionar los que se soportan dentro de la subred de equipos de su tecnología y fabricante. Lo común es que un servicio se soporte sobre varias subredes de fabricantes y tecnologías diferentes y como cada uno de estos sistemas solo posibilita el manejo del fragmento de red de su tecnología y fabricante, al operador se le hace imposible lograr una gestión integrada del servicio extremo a extremo. A su vez, cada sistema de gestión hace un enfoque diferente de los procesos de gestión. Ver Figura 4.1.

La fragmentación de la gestión oscurece la visión global de la red de telecomunicaciones como un todo. Este efecto resulta más grave si se considera que existe un equipamiento que por su simpleza o antigüedad no está representado en ninguno de los sistemas de gestión propietarios.

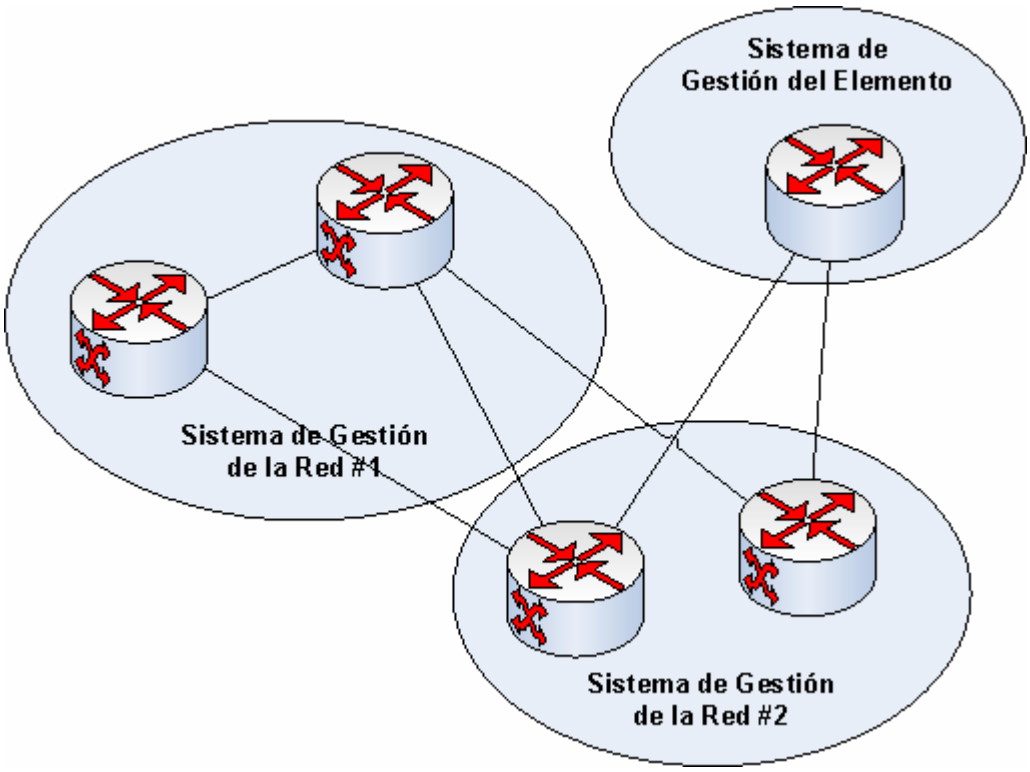


Figura 4.1 Visión fragmentada de la red

Las diferencias del enfoque en cada uno de los sistemas atenta contra el establecimiento de una gestión homogénea de la red, por lo que la empresa se ve obligada a tener una gran cantidad de operadores especializados en las diferentes tecnologías.

La mayoría de los sistemas de gestión con que cuenta la empresa hoy en día, carecen por si solos de los conceptos necesarios para hacer una gestión orientada al negocio. Además, dichos sistemas son cerrados, por lo que la inclusión de estos conceptos depende de los fabricantes, que en el mejor de los casos ponen precios elevados a este tipo de modificaciones personalizadas.

Esta composición compleja de múltiples tecnologías, fabricantes y sistemas aumenta la complejidad de la gestión. No se cuenta con las herramientas necesarias para el nuevo enfoque de gestión. El avance en la infraestructura de software de la empresa depende de sus proveedores de tecnologías. En fin, frena las aspiraciones actuales de la empresa.

4.3 Diseño de la Arquitectura de Integración

Toda plataforma de gestión está conformada por sistemas interconectados que colaboran entre si para un solo propósito, la gestión. La plataforma abandona el marco de las aplicaciones monolíticas para diseñar soluciones que involucren varias aplicaciones. Esto conduce inequívocamente a un escenario de integración de aplicaciones.

La integración es la base para alcanzar la necesaria colaboración de las aplicaciones. Si esta no se lleva a cabo cuidadosamente la plataforma puede que no logre satisfacer los requerimientos de extensibilidad y flexibilidad[8]. Por tanto, es recomendable diseñar la **Arquitectura de Integración** que dará soporte a la plataforma de gestión antes de comenzar a desarrollar la misma.

El diseño preliminar de la Arquitectura de Integración debe satisfacer los requisitos siguientes:

- Especificar el soporte para la automatización de los procesos de negocio que involucren más de una aplicación.
- Especificar las características que deben tener las aplicaciones individuales para una rápida integración dentro de la plataforma.
- Especificar cómo interactuar con los sistemas legados más antiguos dentro de la plataforma.
- Proponer las tecnologías necesarias para la implementación.

El diseño de la solución de integración permite obtener una visión global de la plataforma desde el punto de vista técnico. Esto permite bajar considerablemente los costos y aumentar la velocidad de la integración a la vez que todo el proceso de desarrollo gana en fiabilidad. Introduce un modelo teórico en el que se analiza la evolución y el impacto producido por los cambios. Además, permite evaluar de antemano las tecnologías necesarias para la integración y adicionar su costo al de las aplicaciones.

El inconveniente del diseño prematuro de la solución de integración es que se cuenta con un conjunto muy pobre de requisitos que, por demás, son bastante abstractos. Además, la

[8] Ver el epígrafe dedicado al Problema de la Integración en el Capítulo 1.

experiencia necesaria para diseñar algo nuevo que posteriormente determinará el futuro de una gran infraestructura es insuficiente. El tiempo con que se cuenta para el diseño es bastante escaso pues la empresa necesita contar con la plataforma lo antes posible. Debido a estas condiciones se decidió hacer un diseño de la Arquitectura de Integración basado en patrones.

4.3.1 Automatización de los Procesos de Negocio

Según sugiere la metodología de NGOSS, la plataforma de gestión separa los procesos de negocio de las aplicaciones. De aquí se derivan las interrogantes siguientes: ¿cómo coordinar automáticamente la ejecución de dichos procesos, los cuales abarcan múltiples aplicaciones?, ¿qué entidad está encargada de ello? Este escenario y estas interrogantes se ajustan con el patrón de Integración de Procesos.

Un análisis detallado de las fuerzas que intervienen en el patrón, arrojó un conjunto de valoraciones adicionales sobre el escenario, las cuales son:

- Un proceso de negocio no constituye una lógica estrictamente definida, sino que cada empresa posee una propia.
- La lógica de los procesos cambia con el tiempo con el objetivo de mejorar el funcionamiento de la empresa. En esta evolución la experiencia de los usuarios juega un papel tan importante como los modelos estándares.
- La ejecución de procesos como la provisión de servicios y la gestión de alarmas toma un largo tiempo, por lo que el enfoque sincrónico queda desechado.
- El proceso de negocio abarca varias aplicaciones separadas físicamente, por lo que la probabilidad de un fallo en la comunicación durante la ejecución del mismo es alta.
- Tiene que existir un soporte para operaciones transaccionales de larga duración con el objetivo de mantener la integridad de los datos.

El patrón de Integración de Procesos propone como solución la creación de un **Gestor de Procesos**. El mismo está encargado de controlar la ejecución de los procesos de la empresa definidos en un **Modelo de Proceso** que recibe como entrada dicho componente.

Con la aplicación de la Integración de Procesos, la empresa crea una capa de procesos mantenible, lo que es un requisito imprescindible para la evolución de sus procesos de negocio. Las aplicaciones pueden participar en múltiples procesos como es el caso del Inventario y el Gestor de Fuerza de Trabajo que participan tanto en la Provisión de Servicios como en la Gestión de las Alarmas. La plataforma gana en flexibilidad, escalabilidad y confiabilidad. Permite dar seguimiento a los procesos e informar tanto a los operadores como a los clientes sobre el estado de los mismos. Además, el control centralizado de los procesos permitiría la emisión de reportes que indiquen en donde se encuentran los principales cuellos de botella de la gestión. Esta información tiene un valor incalculable para los directivos en su labor de mejorar cada vez más el rendimiento de la empresa.

El patrón además indica un conjunto de riesgos que deben ser tomados en consideración y se pueden detallar como:

1. Inclinación al uso indiscriminado del Gestor de Procesos.
2. Complejidad de la construcción de un gestor de procesos personalizado.
3. Cuello de botella empresarial

El primer riesgo es soluble si los especialistas de ETECSA consideran en sus análisis las recomendaciones de eTOM. Sin embargo, para los restantes dos riesgos hay que considerar otros factores. El operador, ETECSA, necesita de una solución a corto plazo, por lo cual no es factible la opción de construir un gestor de procesos personalizado; menos aun, si se considera que dicho gestor tiene que poseer características de rendimiento y multiprocesamiento que le permitan soportar la magnitud y complejidad de procesos de un operador de telecomunicaciones sin que ocurran cuellos de botella. Por otro lado, existen productos comerciales para la gestión de procesos con muy buen desempeño, por lo que es mejor adoptar un gestor de procesos profesional en vez de construir uno propio.

Uno de los refinamientos del patrón Integración de Procesos[9], sugiere a Microsoft Biztalk Server 2004 como gestor de procesos para la implementación de una capa de integración de procesos. Una revisión más detallada de las fuentes [Microsoft04] proporciona un caso de

[9] Implementing Process Integration with BizTalk Server 2004 [Townbridge04]

estudio hecho por Microsoft donde se analiza la implementación de los procesos de eTOM utilizando Biztalk. Teniendo en cuenta esta información y el análisis de las capacidades de Microsoft Biztalk Server como gestor de procesos[10], se adoptó esta tecnología para la Capa de Integración de Procesos de la plataforma.

En resumen, para dar soporte a los procesos de negocio de la Plataforma de Gestión de ETECSA se propone la creación de una Capa de Integración de Procesos con un Gestor de Procesos basado en Microsoft Biztalk Server 2004 como se muestra en la Figura 4.2.

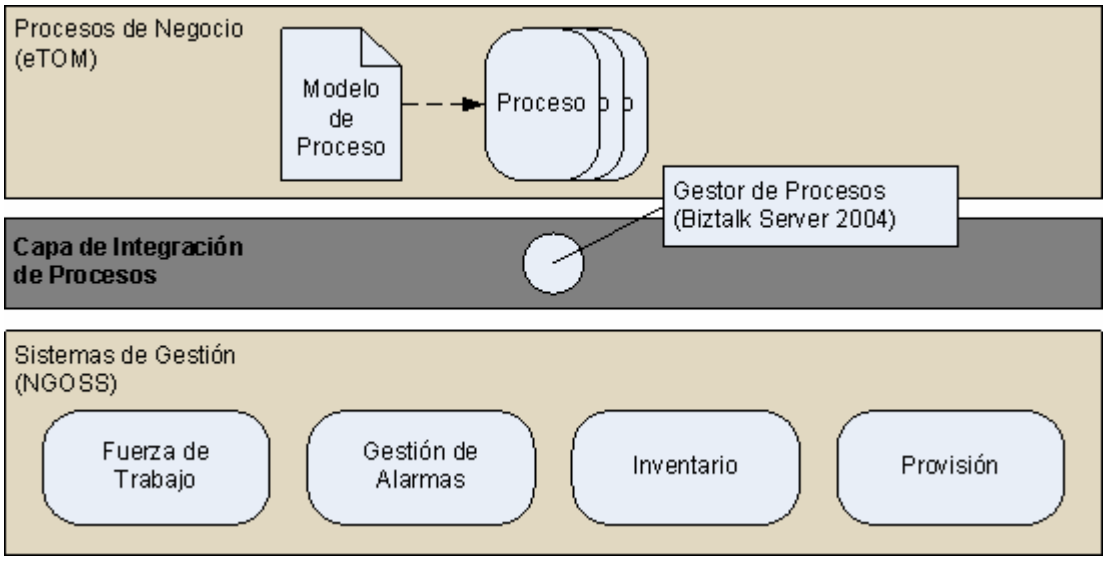


Figura 4.2 Creación de una Capa de Integración de Procesos

4.3.2 Orientación de las Aplicaciones a Servicios

Toda arquitectura de integración debe abaratar y agilizar el proceso de integración. Dicho en otras palabras, debe agilizar y abaratar la inclusión de nuevas aplicaciones y la automatización de las interacciones entre las aplicaciones dentro de la plataforma. Un aspecto clave para ello es la especificación de la comunicación de las aplicaciones entre si y con el gestor de procesos.

Con el objetivo de garantizar la comunicación entre las aplicaciones, TNA[11] propone el uso de interfaces contractuales (NGOSS Contract). Esta idea coincide con el patrón Integración Orientado a Servicios.

[10] Ver el epígrafe dedicado a Microsoft Biztalk Server 2004 en el Capítulo 3.
[11] Technology Neutral Architecture. Ver el epígrafe dedicado a NGOSS en el Capítulo 1.

Asumiendo el patrón de Integración Orientada a Servicios se satisfacen los deseos de la empresa de mantenerse acorde a la línea establecida por NGOSS. Al mismo tiempo, se garantiza la interoperabilidad entre los diferentes paradigmas tecnológicos (Java, .Net, Pitón, etc), lo cual es muy importante para conectarse a los diversos sistemas legados. Además, el uso de los servicios posee otro beneficio, el acople débil. Esto reduce significativamente la dimensión del impacto producido por los cambios, permitiendo que cada uno de los sistemas evolucione sin que colapse la plataforma.

El patrón de Integración Orientada a Servicios hace un análisis del acoplamiento entre las aplicaciones desde varios puntos de vista. La solución propuesta por el patrón logra un balance tal de las fuerzas que intervienen en el acoplamiento, que los Servicios Web son considerados el paradigma actual para la producción de sistemas débilmente acoplados.

Los servicios no imponen ninguna restricción en cuanto al contenido semántico de los mensajes que se intercambian los componentes. Ello permite la asimilación de la terminología expuesta por NGOSS SID (NGOSS Shared Information/Data Model), abriendo el camino para la estandarización de la terminología y los datos. En un futuro esto puede ser un factor clave para la integración interempresarial.

En cuanto a las tecnologías a utilizar, los Servicios Web permiten escoger entre un amplio arco de tecnologías esparcidas por casi todos los ambientes de programación. Microsoft recomienda el uso de XML Web Services como base para la integración [Microsoft04]. Tomando en cuenta el análisis efectuado sobre las diversas tecnologías de implementación de Servicios Web en plataformas Windows, se determinó que los ASP. Net Web Services son suficientes para la mayoría de los casos de uso.

El resto de los casos de uso se corresponden con escenarios en los cuales el tamaño de los mensajes superaba 1MB y/o el tiempo de respuesta superaba los 5 minutos. Las pruebas realizadas a las diferentes tecnologías demostraron que las versiones actuales de ASP.Net Web Services y WSE no brindaban los requerimientos de fiabilidad necesarios. Para estos casos, se propuso utilizar una Capa Intermedia Orientada a Mensajes soportada sobre Microsoft Message Queuing (MSMQ). En el futuro, cuando las tecnologías de Servicios Web ofrezcan mejores posibilidades operacionales, dicha capa puede ser sustituida.

En resumen, la comunicación entre las aplicaciones de la plataforma se basa fundamentalmente en el uso de Servicios Web implementados en ASP.Net. En el caso de las excepciones explicadas anteriormente se utiliza una Capa Intermedia Orientada a Mensajes soportada sobre MSMQ. Todo lo cual se muestra en la Figura 4.3.

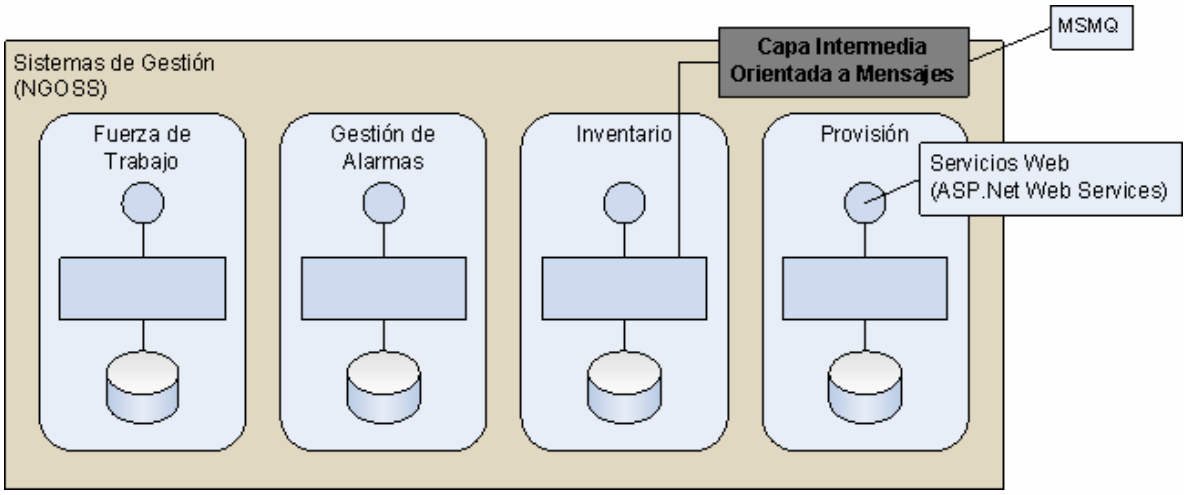


Figura 4.3 Comunicación con los sistemas de gestión

4.3.3 Integración con los Sistemas de Gestión Legados

Por sistemas legados se entiende el conjunto de sistemas con que cuenta la empresa y que no están acorde con las líneas que impone la plataforma. A pesar de ello, la función que cumplen dichos sistemas y las informaciones que ellos manejan tiene una gran importancia para la empresa. Algunos de los sistemas legados llegan a estar tan arraigados a la rutina de trabajo, que una interrupción brusca de sus servicios podría significar el caos.

Efectuar la integración con los sistemas legados es una de las tareas más complejas de la plataforma. En ella se combinan la heterogeneidad de los modelos y las tecnologías. Muchas de estas aplicaciones existentes nunca fueron preparadas para la interoperabilidad y por tanto, los mecanismos para lograr la misma son costosos e implican riesgos que deben ser tomados en consideración.

4.3.3.1 Integración de los Datos

Uno de los problemas concretos lo constituye el problema del Inventario de la Red. En el sistema de inventario se combinan los datos de todos los equipos, la configuración física y lógica de las conexiones entre estos y los servicios soportados por la red. La carga de la información es mixta, una parte se hace manualmente, mientras lo otra se obtiene

automáticamente de los sistemas de gestión de red legados. Aumentando la automatización de la carga de información, se logra un mayor nivel de actualización de los datos en el Inventario, se reduce el personal dedicado a la cargado manual y se disminuyen los errores humanos.

Una situación similar le ocurre al Gestor de Alarmas en el proceso de detección de las alarmas. En este caso, muchos de los sistemas de gestión legados son capaces de detectar automáticamente las alarmas que ocurren en el equipamiento. Sin embargo, la empresa requiere de un personal que se mantenga supervisando e introduzca los datos manualmente en el Gestor de Alarmas. Mediante la integración del Gestor de Alarmas y los sistemas de gestión legados, se gana en velocidad a la hora de responder a una alarma, se reduce considerablemente el trabajo de los supervisores y se disminuyen los errores humanos.

Ya sea en el Inventario o en Gestor de Alarmas, la empresa necesita tener una vista unificada de los datos que se encuentran dispersos en varios sistemas de gestión legados. Este escenario es propicio para la aplicación de una Capa de Agregación de Entidades. La cual, además de proveer la necesaria vista unificada, permite establecer un modelo de datos único para toda la capa de gestión empresarial simplificando el futuro desarrollo de aplicaciones.

El patrón clásico de Agregación de Entidades, sugiere agregar los datos mediante referencias a los mismos en cada uno de los gestores. Sin embargo, en este caso se optó por efectuar una copia de los datos de interés hacia una base de datos intermedia.

La copia de los datos conduce a problemas de transferencia de datos y de sincronización entre bases de datos, los cuales no están presentes en la solución estándar. A pesar de ello, la existencia de una base de datos intermedia permite independizar las aplicaciones de gestión empresarial de las aplicaciones de gestión legadas. Esta independencia permite a la empresa hacer cambios en su infraestructura de gestión legada sin afectar las capas superiores directamente; permite incluir la información que no pertenece a ningún sistema de gestión legado y finalmente, como esta base de datos contiene los datos homogeneizados de toda la red, es ideal para la elaboración de reportes o como base para otras aplicaciones empresariales.

La Agregación de Entidades presupone una resolución completa de la disonancia semántica. Como cada sistema de gestión legado posee su propio modelo de representación de los datos, este proceso de copia hacia la base de datos intermedia deberá incluir una lógica de transformación de un modelo de datos a otro.

El patrón establece que para efectuar la integración de datos es preciso efectuar una conexión física a los datos de los sistemas legados, lo cual se corresponde con el patrón de Integración de Datos. El acceso físico a los datos, ofrece mejores rendimientos que otros tipos de integración. El rendimiento juega un papel muy importante ya que trata de sincronizar grandes volúmenes de datos.

El patrón de Integración de Datos llama la atención sobre los riesgos que se corren al saltar la capa de negocio de la aplicación. Al no estar presente una lógica que controle el acceso a los datos, una operación errónea puede corromper la información. Este riesgo tiene una alta probabilidad de ocurrir, pues los sistemas de gestión legados pertenecen a un conjunto de fabricantes que normalmente no proveen una documentación que especifique la estructura interna de sus sistemas. Ante esta expectativa, se resolvió utilizar un enfoque unidireccional de solo lectura, en vez del clásico enfoque bidireccional. A medida que ETECSA adquiera nuevos OSS o versiones de los existentes acordes con las recomendaciones del TMF expresadas en su iniciativa NGOSS, se podrá efectuar una integración a nivel de capa de negocio con cada sistema legado, permitiendo un intercambio bidireccional seguro.

Mediante el uso de una Capa de Agregación de Entidades en conjunto con la Integración de Datos, se obtiene una vista unificada y homogénea de los datos de los sistemas de gestión legados a nivel empresarial. La tecnología que ofrece las mejores características[12] para la implementación de ambos patrones es Microsoft SQL Server Data Transformation Services. La ubicación de la Capa de Agregación de Entidades dentro de la arquitectura se muestra en la Figura 4.4.

[12] Ver el epígrafe dedicado a Microsoft SQL Server Data Transformation Services en el Capítulo 3.

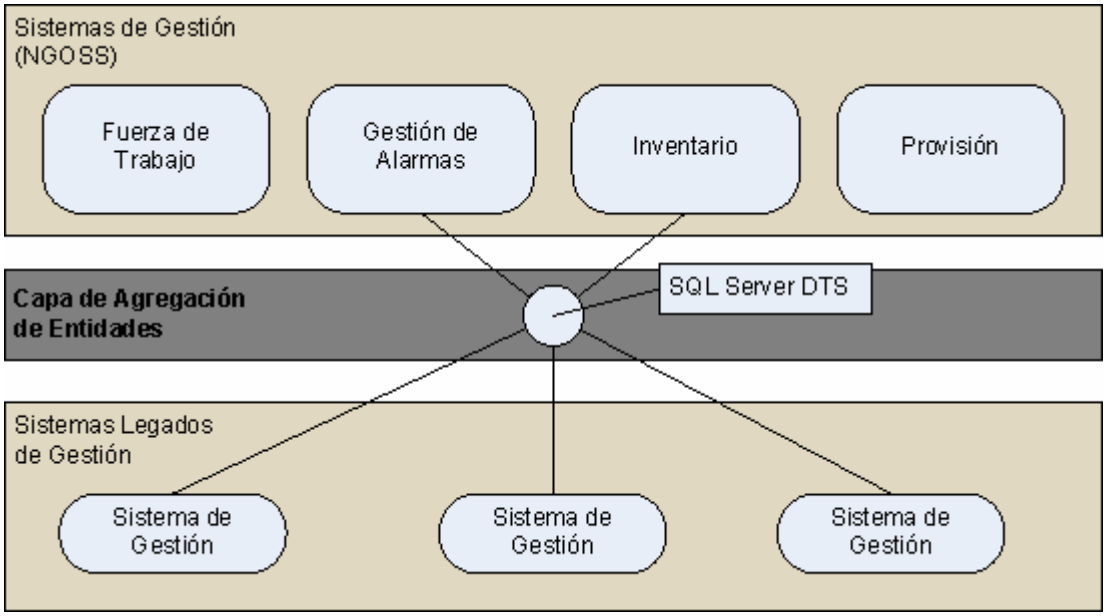


Figura 4.4 Creación de una Capa de Agregación de Entidades

4.3.3.2 Integración de las Operaciones

Así como los datos que están presentes en los sistemas de gestión legados son transportados hacia la capa de gestión empresarial, las operaciones que tienen lugar en la capa de gestión empresarial tienen repercusión sobre los sistemas de gestión legados. Los OSS provistos por los fabricantes, son los sistemas que en un final están encargados de la configuración física de los equipos y del monitoreo real de los fallos. Por lo tanto, ellos son el soporte de más bajo nivel para la automatización de los procesos de negocio.

Una idea inicial puede ser conectar los OSS directamente al Gestor de Procesos. Sin embargo, esta idea se enfrenta a tres problemas.

1. La gestión de la red está fragmentada en varios de estos sistemas. El Gestor de Procesos tiene que determinar cuál de ellos está encargado de efectuar determinada operación. Si dicha operación abarca más de un sistema, es preciso determinar que parte le corresponde a cada uno.
2. Los sistemas legados no comparten ni modelo, ni una interfaz homogénea con la plataforma. Los modelos de proceso deben modificados para ajustarlos a las particularidades de cada OSS. Los procesos de negocio pierden su generalidad y se tornan más complejos.

3. Algunos de los OSS nunca fueron pensados para interoperar y por tanto la integración es demasiado costosa o imposible, en dichos casos se prefiere conservar la interacción manual.

Si el Gestor de Procesos tuviera que interactuar directamente con los sistemas de gestión legados en la modelación de los procesos de negocio, se tendrían lógicas demasiado complejas, plagadas de casos particulares y excepciones, difíciles de comprender y modificar. Esto se aleja de los modelos propuestos por eTOM.

Por tanto, se determinó limitar la interacción del gestor de procesos solo a las aplicaciones de la capa de gestión empresarial, en donde existe una visión unificada de la red a través de los sistema de Inventario y Gestión de Alarmas. Estos sistemas están encargados de conciliar los datos de los OSS legados y presentarlos en una sola vista, por lo que son los sistemas más apropiados para controlar la lógica de interacción con los sistemas legados.

Para mantener una lógica simple, es preciso desacoplar las aplicaciones de la capa de gestión empresarial de los sistemas de gestión legados. Con este objetivo se decidió aplicar un refinamiento del patrón Broker, el Broker de Mensajes.

A través del Broker de Mensajes, los sistemas empresariales de gestión envían mensajes hacia los sistemas legados ordenándoles ejecutar una operación. La secuencia de pasos es como sigue: el Broker recibe dicho mensaje, determina cuál es el sistema receptor, transforma los datos hacia el modelo de dicho sistema, establece la comunicación, recoge la respuesta, la transforma y la envía de vuelta.

En caso que el sistema sólo permita la interacción manual, entonces el Broker envía el mensaje hacia un Sistema de Notificación, el cual está encargado de elaborar una orden de trabajo para el operario y hacérsela llegar por un medio de comunicación (correo electrónico). El operario realiza la tarea manualmente y envía la respuesta hacia el Broker que a su vez la reenvía hacia el sistema de gestión empresarial.

Desde el punto de vista de las conexiones, el patrón más apropiado a utilizar es la Integración Orientada a Servicios. La causa fundamental de esta decisión lo constituye el

alto grado de interoperabilidad necesario, ya que los sistemas de gestión legados pertenecen a las más diversas tecnologías.

La principal dificultad a la hora la aplicar la Integración Orientada a Servicios es que, al ser un paradigma bastante moderno, la mayoría de los sistemas carecen de las estructuras necesarias. A pesar de esto, la orientación a servicios es el paradigma que defiende NGOSS para la comunicación, por lo que no hay que perder de vista que los fabricantes, con el objetivo de alinearse a la corriente mundial, comiencen a proveer este tipo de interfaces en las subsecuentes versiones de sus software. Por tanto, sería una decisión estratégica de ETECSA establecer las bases para este tipo de comunicación.

Aplicando ambos patrones, el Broker de Mensajes y la Integración Orientada a Servicios, se propuso una solución para la integración de las operaciones. Dicha solución se muestra en la Figura 4.5.

Como tecnología para la implementación del Broker de Mensajes, se determinó utilizar Microsoft Biztalk Server 2004 atendiendo a lo sugerido por el patrón en uno de sus refinamientos[13] y al análisis previo de sus capacidades[14]. La arquitectura de mensajería de Biztalk posee capacidades nativas de enrutamiento y transformación de los mensajes, ambas requeridas en este escenario. Además, las posibilidades de Biztalk de elegir entre varios mecanismos de comunicación con las aplicaciones, permite resolver los casos en los cuales no sea posible utilizar la Integración Orientada a Servicios.

[13] Implementing Message Broker with BizTalk Server 2004 [Townbridge04]
[14] Ver en el Capítulo 3 el epígrafe dedicado a Microsoft Biztalk Server 2004.

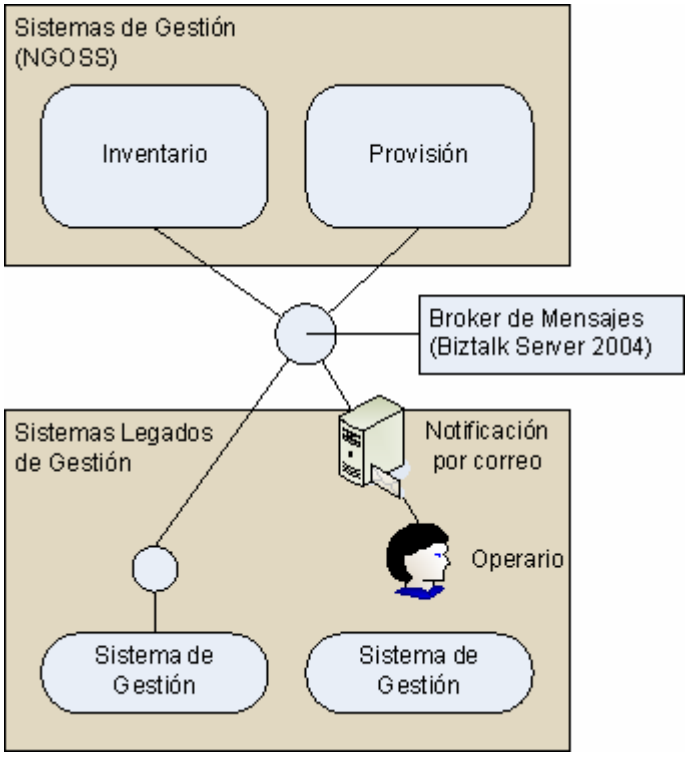


Figura 4.5 Broker para la comunicación con los sistemas legados

4.4 Mapa de la Arquitectura de Integración

Después de analizar por separado cada uno de los requisitos de la plataforma, determinar soluciones a los problemas desde el punto el punto de vista teórico y proponer las tecnologías para la implementación; se obtuvo el mapa de la Arquitectura de Integración, el cual se muestra en la Figura 4.6.

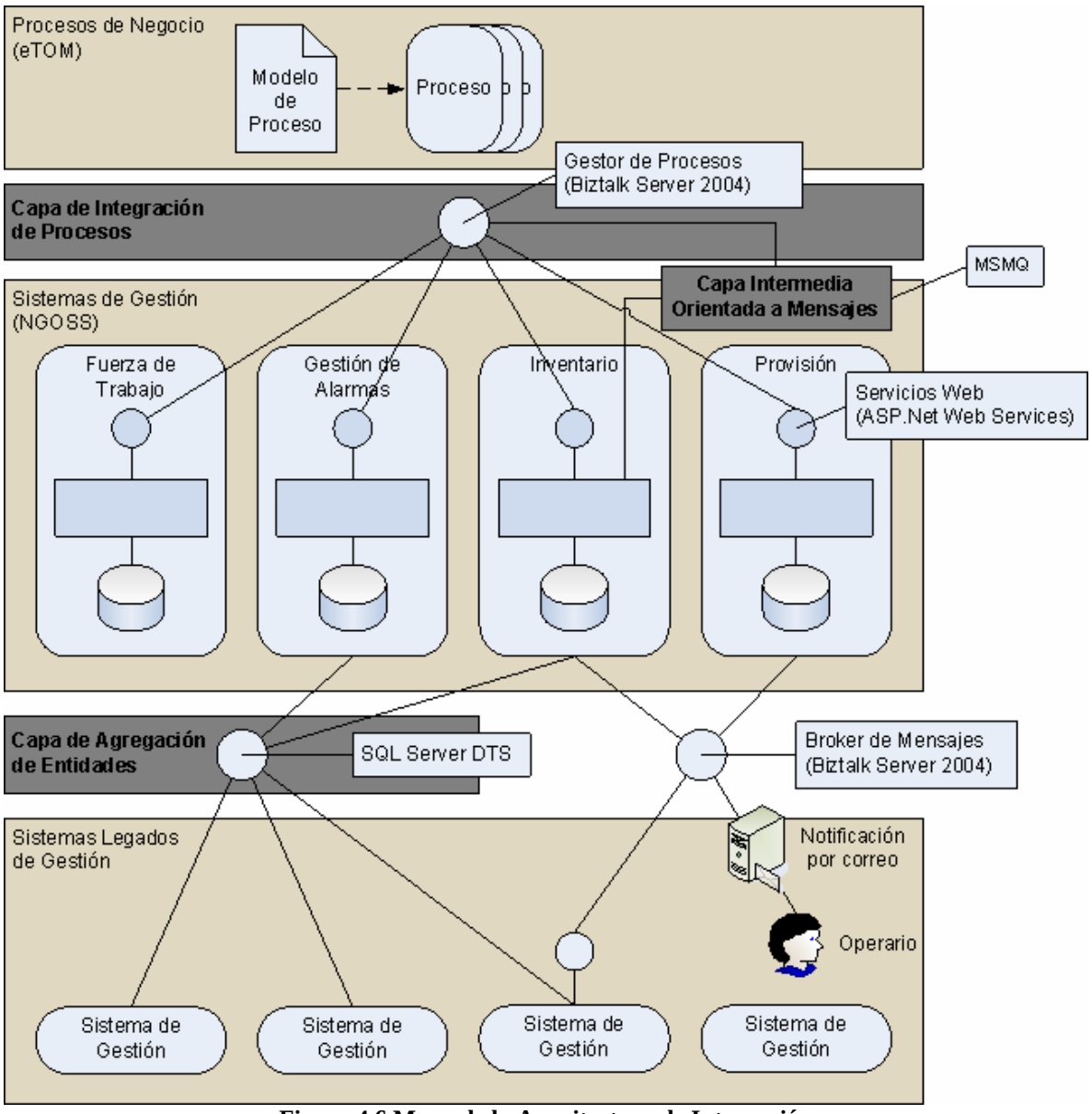


Figura 4.6 Mapa de la Arquitectura de Integración

El mapa expresa el diseño de la Arquitectura Integración como una combinación de patrones de integración. Se observan claramente las relaciones entre cada una de las partes de la arquitectura, en las cuales se ubican las principales tecnologías que serán usadas para su implementación.

4.5 Análisis de los resultados

El diseño de la Arquitectura de Integración obtenido satisface cada uno de los requisitos impuestos por ETECSA y al mismo tiempo está acorde a los paradigmas propuestos por

NGOSS. Este resultado tiene un alto valor para la empresa en su estrategia para la construcción de una plataforma de gestión a la altura de estos tiempos.

Los análisis efectuados y las solución propuesta resultaron útiles para la elaboración de un conjunto de recomendaciones en la compra e implementación de nuevos sistemas de gestión. Estas recomendaciones ya se están aplicando en la evaluación de una suite del Sistema Gaïa que se pretende adquirir.

Por otro lado, en un futuro próximo, se comenzará la implementación de una nueva versión del Inventario de la Red y la construcción del Gestor de Alarma, el Gestor de la Fuerza de Trabajo y el Sistema de Provisión. Estos sistemas harán uso de la arquitectura obtenida para la automatización de los procesos de Provisión y Mantenimiento y la integración con los OSS legados.

CONCLUSIONES

- Los patrones de diseño resultan muy efectivos en la solución de problemas complejos de diseño, fundamentalmente durante su etapa temprana.
- Los patrones de diseño permiten hacer una evaluación de las capacidades reales de las tecnologías para solucionar determinada clase de problemas.
- El cluster de patrones de integración hace un estudio sistémico de la problemática de la integración que puede ser utilizado por los desarrolladores para el diseño de arquitecturas.
- Existen tecnologías y herramientas profesionales suficientes para una rápida implementación de los patrones de integración en un entorno Windows.
- La metodología utilizada permite obtener el diseño de la Arquitectura de Integración requerida y hacer una propuesta de las tecnologías para su implementación.
- El diseño obtenido no sólo satisface los requisitos impuestos por ETECSA sino que además está acorde con los paradigmas propuestos por NGOSS.

RECOMENDACIONES

- Generalizar el uso de patrones en la etapa temprana del diseño.
- Incluir el estudio de los patrones de diseño como parte del currículo de Ciencias de la Computación y otras especialidades de la informática.
- Generalizar hacia una metodología, el análisis sistémico de los patrones efectuado en el Capítulo 2.
- Utilizar las recomendaciones de NGOSS para la concepción, evaluación y compra de nuevos Sistemas de Soporte a la Operación y como una guía para la construcción de la Plataforma de Gestión de ETECSA.
- Implementar el diseño la Arquitectura de Integración para que de soporte a corto plazo al Sistema de Inventario de la Red, al Gestor de Alarma, al Gestor de la Fuerza de Trabajo y al Sistema de Provisión.

BIBLIOGRAFÍA

[Alexander79] Alexander, C. (1979). The Timeless Way of Building.

[Biztalk04] Microsoft Corporation (2004). Biztalk Server 2004 Documentation.

[Booch98] Booch, Graddy (1998). Análisis y Diseño Orientado a Objetos con Aplicaciones. Addison Wesley, segunda edición 1998.

[Box03] Box, Don (2003). "SOAP: Service Oriented Architecture and Programming". MSDN TV Conference. 2003.

[Coplien00] Coplien, James O. (2000). Software Patterns. Bell Laboratories, The Hillside Group.

[Hohpe03] Hohpe, Gregor (2003). Enterprise Integration Patterns.

[Microsoft03] Microsoft Corporation (2003). Accelerate OSS/BSS agility by migrating to a new generation OSS based on XML and Web services.

[MSDN04] Microsoft Corporation (2003). Microsoft Developer Network Documentation, Julio del 2003.

[MSMQ03] Microsoft Corporation (2003). Message Queuing Help.

[Patternshare05] Patternshare (2005). Catálogo Online. http://www.patternshare.com

[Peter03] Peter, L. (1986). The Peter Pyramid. New York, NY: William Morrow.

[Rodríguez03] Rodríguez, Jesús M (2003). Plataforma GENIO basada en patrones sobre Microsoft.NET.

[Siegel00] Siegel, Jon (2000). CORBA 3 Fundamentals and Programming. John Wiley & Sons, Inc. 2000.

[SQL00] Microsoft Corporation (2000). SQL Server Books Online.

[Strassner03] Strassner, John (2003). TMF White Paper on NGOSS and MDA version 1.0. TeleManagement Forum

[TMF04] TeleManagement Forum (2004).The NGOSS Technology-Neutral Architecture. NGOSS Release 4.0.

[TMF05] TeleMenagement Forum (2005). TMF Frecuently Asked Questions. http://www.tmforum.org

[Trowbridge03] Trowbridge, David (2003). Enterprise Solution Patterns Using Microsoft .NET, Version 2.0.

[Trowbridge04] Trowbridge, David (2004). Integration Patterns.

[W3C04] W3C Working Group. Web Services Glossary. http://www.w3.org