CAPÍTULO 2
Evolución de los Sistemas Distribuidos

Patricia Bazán

Los sistemas distribuidos concebidos como aquellos que funcionan como si se tratara de un sistema único, no hacen ninguna presuposición acerca de la variedad funcional de sus componentes ni de cómo estos componentes se comunican. De esta manera las variantes que existen han dado origen a sistemas distribuidos de distinta índole y que van de la no-distribución (sistemas monolíticos) hacia la distribución completa (servicios orquestados por procesos de negocio).

En este capítulo se analiza esta evolución tanto desde el punto de vista tecnológico que sin dudas se ve afectado por la evolución de las comunicaciones, plataformas y hardware.

El capítulo se organiza de la siguiente manera: la Sección 1 presenta el concepto de sistema monolítico y sus mejoras ante la aparición de distintos componentes funcionales. En la Sección 2 se describen los sistemas Cliente/Servidor y su evolución hacia sistemas de n-capas. En la sección 3 se analizan los sistemas distribuidos con objetos como antecesores a las arquitecturas orientadas a servicios presentadas en la Sección 5. Mientras tanto, en la Sección 4 se describen las distintas alternativas de integración de aplicaciones. En la Sección 6 se analizan a los procesos de negocio como consumidores de Servicios. Finalmente en la Sección 7 se arriban a algunas conclusiones.

2.1. Sistemas Monolíticos

Un sistema de información monolítico es aquel que se concibe como un único elemento funcional donde sus prestaciones se ofrecen a través de una sola pieza de código que resuelve tanto la presentación al usuario (interface), como el acceso a los datos (trata con operaciones de entrada/salida) y a su vez resuelve la lógica algorítmica del problema que aborda.

Estos sistemas de información se construían en un único lenguaje de programación, sobre un único computador y con un único sistema operativo como plataforma. La comunicación con el usuario y también con el programador, era a través de terminales de caracteres que responden únicamente a comandos lanzados por consola. Su ubicación en el tiempo se remonta a los años ‘70.

Una versión mejorada de los sistemas de información así concebidos, pudo aportarse con las técnicas de programación estructurada, los lenguajes de programación y las metodologías

de programación modular introducidas por Edsger Dijkstra. En este sentido, los sistemas de información comenzaron a ser concebidos como un conjunto de componentes o niveles de complejidad, aunque permanecían ejecutándose en una misma computadora y por lo tanto podemos considerarlos no-distribuidos.

La Figura 16 la evolución marca que el primer paso fue reconocer un componente funcional que resuelve el acceso a los datos (con la aparición de los DBMS-DataBase Management Systems), para luego identifica un componente de manejo de interface de usuario [Weske, 2008].

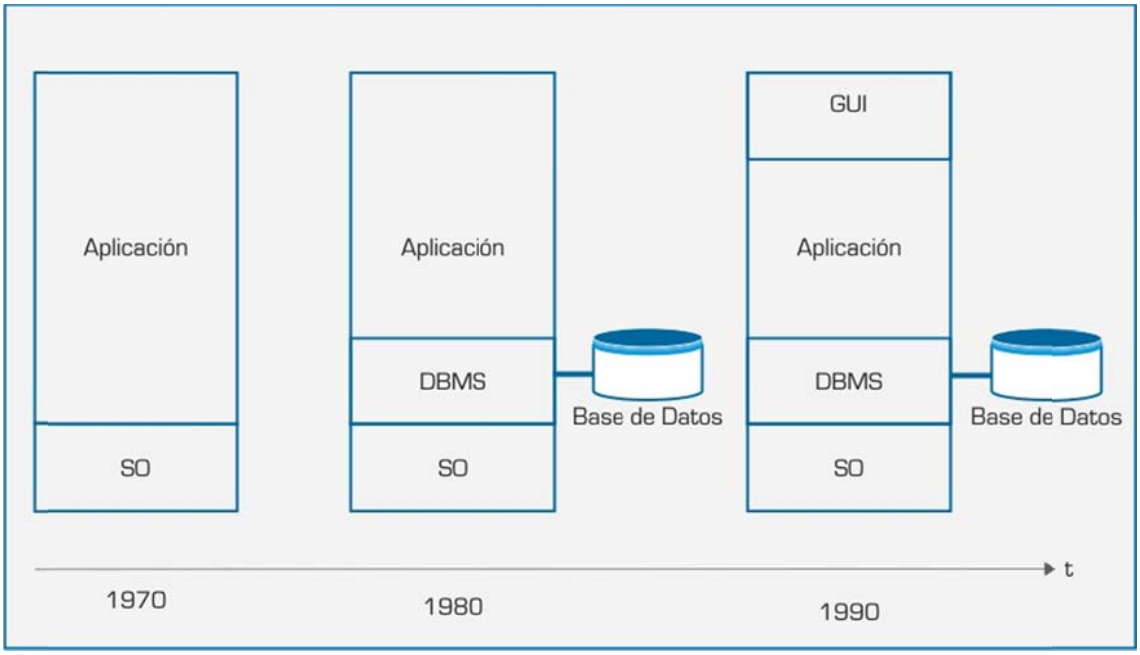

Fig. 16. Evolución tecnológica de los sistemas de información

2.2. Cliente/Servidor y su evolución a n-capas

La distribución funcional entre lógica de aplicación propiamente dicha y lógica de acceso a datos vislumbrada en la década del 80, condujo a concebir la idea que estos dos componentes podría residir en computadoras diferentes aprovechando las consultas SQL que podían interpretar los DBMS, junto con la gran proliferación de PCs sobre LAN (Local Area Network), que se estaba dando en esos tiempos.

Este escenario dio origen a la primera versión de sistema distribuido, desde el aspecto funcional, que fue el denominado Cliente/Servidor o arquitectura de 2 capas (En el Capítulo 4 se analizará en detalle esta arquitectura).

La arquitectura Cliente/Servidor o arquitectura de 2 capas clásica, se sustenta en la idea que existe una componente Cliente que resuelve la lógica de la interface con el usuario. El código de este componente se escribe generalmente en un lenguaje visual de cuarta generación (Delphi, VisualBasic, PowerBuilder). La capa 2 sería un componente Servidor que resuelve la lógica de acceso a los datos, al menos con la gestión que hace los mismos un DBMS.

En esta arquitectura, la lógica de la aplicación no cuenta con un componente físico específico para su ejecución y despliegue, sino que es absorbida por algunas de las dos componentes antes mencionadas: Cliente o Servidor, tal como se muestra en la Figura 17.

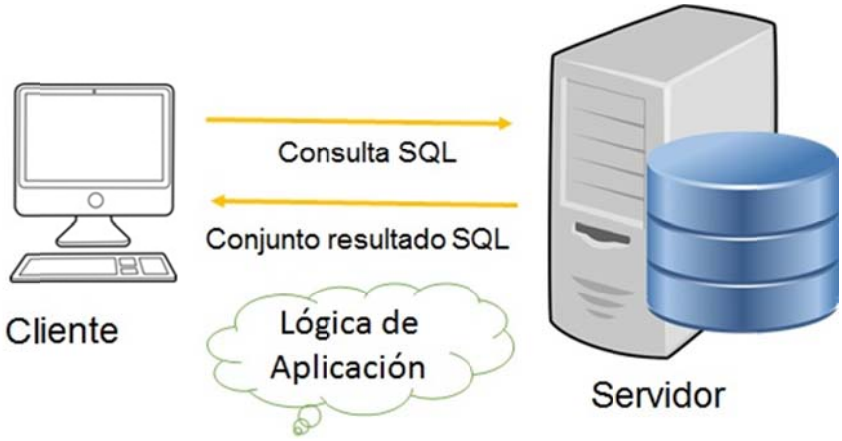

Fig. 17. Sistema distribuido en 2 capas o Cliente/Servidor clásico

Este modelo de sistema distribuido posee ventajas y desventajas.

Ventajas

- Aprovechamiento del poder de las PC reduciendo costos
- Permite que el procesamiento resida cerca de la fuente de datos reduciendo el trafico de red
- Facilita el uso de GUI
- Acepta el desafío de los sistemas abiertos
- Favorece el diseño modular

Desventajas

- Un desbalanceo de tareas entre cliente y servidor puede provocar cuellos de botella
- El desarrollo de aplicaciones es más complejo creciendo también la complejidad de las herramientas de desarrollo y programación

La desventaja fundamental que llevó a buscar evolucionar hacia un modelo superador lo constituyó sin dudas el desbalanceo de carga entre cliente y servidor, lo cual se vio acompañado por la instauración cada vez más firme de la Internet como la mejor solución a la no disponibilidad de recursos de cómputo y la globalización de los sistemas de información.

Por su parte, el modelo Cliente/Servidor clásico se transformó en poco escalable (capacidad de crecimiento) tanto a nivel vertical (aumento de cantidad de clientes y por ende de requeri-

mientos al servidor), como horizontal (un solo servidor accesible a través de una LAN, ya no era suficiente, se requería acceso a través de WAN).

Surge así el modelo de 3 capas donde, básicamente, se le otorga a la capa media (lógica de la aplicación) una plataforma de ejecución propia. El más claro ejemplo de este tipo de modelos lo constituye la tecnología Web. De este modo, y dado que el componente medio posee su propia plataforma de ejecución y que se cuenta con un modelo Cliente/Servidor, es que la capa media puede ser una sola o varias - siendo todas estas alternadamente cliente y servidor de la siguiente - y obteniendo la arquitectura n-capas, como se muestra en la Figura 18.

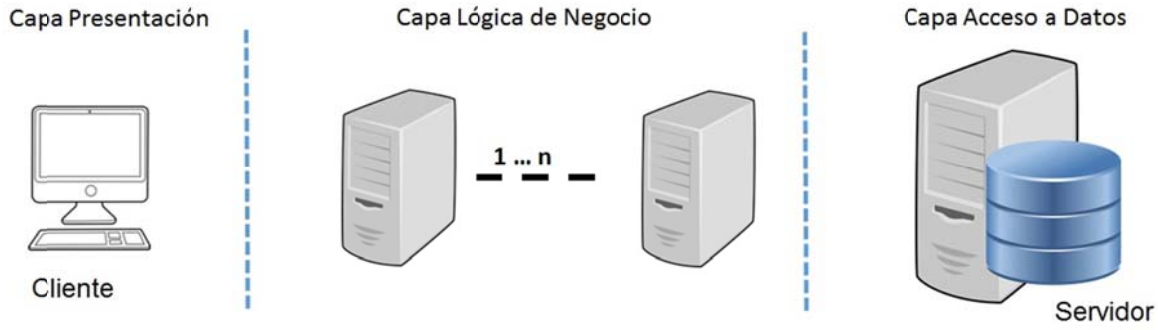

Fig. 18. Sistema de Información en N capas

Tecnología Web: un caso particular de Cliente/Servidor

La tecnología Web es aquella hace uso de Internet para construir sistemas de información. En este sentido, dichos sistemas de información se construyen haciendo uso de los protocolos y mecanismos de comunicación provistos por Internet, esto es: acceder a recursos a través de una identificación única (URL), obtener dichos recursos con un protocolo de transferencia de archivos (FTP/HTTP) y mostrar los resultados al usuario (Navegador o Browser).

Web Browser: componente cliente para visualizar un documento HTML

Web Server: componente que contiene páginas HTML que envía a los clientes que los visualizan en el Web Browser

Web Site: estructura de directorio para almacenar las páginas HTML y otros archivos.

La Figura 19 muestras una solución en tecnología Web simple que se compone por un Web Browser que interpreta páginas HTML (Browser), obtenidas por HTTP de un Web Server.

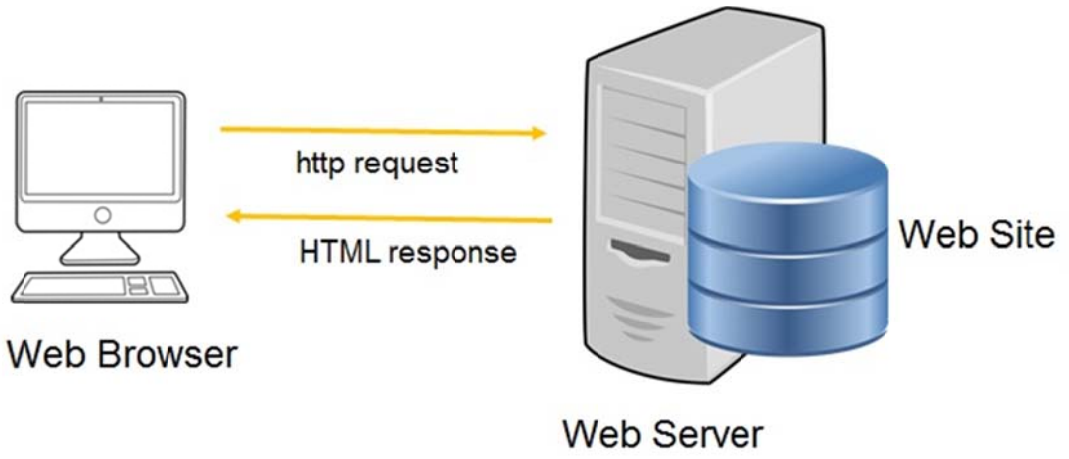

Fig. 19. Tecnología Web Simple

Tecnología Web en 3 capas: Web Server Dinámico

El componente Web Server en sistemas distribuidos de tecnología Web, provee un tipo de servicio que se limita a recibir peticiones por HTTP y responder con un conjunto de registros (archivo) estructurado como HTML.

Este escenario resulta insuficiente para construir sistemas de información debido a: 1- ausencia de un componente que permita acceder a los datos y mantenerlos persistentes y 2- ausencia de un componente que ejecute código para producir transformaciones en dichos datos.

Surge así lo que damos en llamar *web server dinámico*, es decir, aquel que es capaz de desencadenar cómputo que se encuentra codificado dentro de las páginas HTML que administra, tal como muestra la Figura 20.

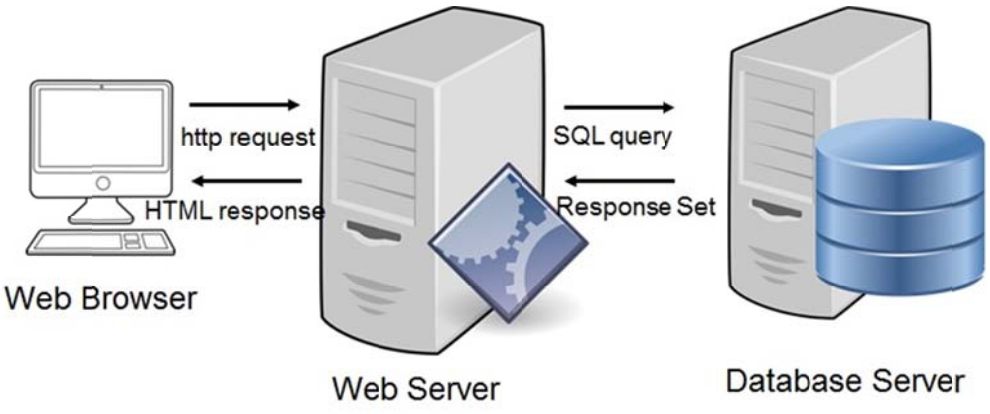

Fig. 20. Tecnología basada en Web en 3 capas

En el Capítulo 5 retomaremos este tipo de tecnología y profundizaremos en ella.

2.3. Sistemas Distribuidos con Objetos

A medida que fue aumentando la complejidad de las soluciones distribuidas fue necesario incorporar conceptos, metodologías y buenas prácticas en la construcción de software y llevarlas al entorno distribuido.

El concepto de *objeto* en el sentido clásico de la Programación Orientada a Objetos y que tiene sus raíces principales en el lenguaje Smalltalk, se mantiene a nivel de diseño de aplicaciones pero cambia algunas características cuando se lo lleva a un entorno distribuido.

Un *objeto distribuido* es un *objeto* (componente funcional con estado interno, comportamiento, heredable y encapsulado), que por estar en un entorno distribuido también debe ser:

- Transaccional: los cambios de estado que produce se deben ejecutar completamente o no ejecutarse.
- Seguro: debe evitar vulnerabilidades que corrompa el estado del sistema.
- *Lockeable*: debe fijar un cerramiento que garantice su ejecución correcta ante accesos concurrentes.
- Persistente: su estado interno debe conservarse más allá de su ciclo de vida.

Esta definición de objeto distribuido nos conduce a la necesidad de contar con una infraestructura que facilite la comunicación entre este tipo de componentes, logrando a su vez que logren esta comunicación a través de distintas plataformas de ejecución, sistemas operativos y lenguajes de programación.

Así surge el estándar CORBA (Common Object Request Broker Architecture) [5] [2.3], definido por la OMG (Object Management Group) y que surge de la iniciativa de un consorcio de empresas interesadas en establecer una arquitectura interoperable sobre la base del paradigma orientado a objetos. CORBA se constituye así en un ejemplo de sistema distribuido basado en objetos.

Entre los elementos distintivos del estándar - y que se puede decir que sentaron las bases para lo que se conoce actualmente como *web services* - encontramos:

- Un lenguaje de especificación de interface, IDL Interface Definition Language, que define los límites de las componentes o sea las interfaces contractuales con los potenciales clientes.
- Un *bus* de objetos u ORB (Object Request Broker) que comunicar objetos independientes de su locación. El cliente se despreocupa de los mecanismos de comunicación con los que se activan o almacenan los objetos servidores.

Provee un vasto conjunto de servicios de middleware distribuido y tiene claramente un mecanismo de comunicación mucho más complicado que los clásicos RPC y MOM.

5 http://www.corba.org/

2.4. Integración de Aplicaciones

Más allá de los esfuerzos por construir estándares para mejorar la interoperabilidad y de la existencia de plataformas Web mucho más accesibles al programador y al usuario, el desarrollo de aplicaciones distribuidas a gran escala, seguía adoleciendo de problemas de integración.

La idea de construcción de aplicaciones integradas permitió que las organizaciones desarrollen software que resuelva cada parte de su negocio y se integre con aplicaciones que gestionen la parte administrativa de dicho negocio. En este sentido la idea de integración de aplicaciones comienza a cobrar un sentido muy relevante.

En este contexto surgen los sistemas ERP (Enterprise Resource Planning) que proveen variada funcionalidad integrada por un único repositorio de datos.

No se tardó demasiado en intentar agregar valor a los desarrollos de las organizaciones aportando sistemas de CRM [6] (Customer Relationship Management) o sistemas de DataWareHouse[7] que requieren algún tipo de integración con los sistemas de índole operativa o propia del negocio.

Esta integración se enfoca principalmente en la integración vía el modelo de datos. Este tipo de integración ha sufrido una evolución e incluye diversas variantes que se describen a continuación.

Integración Punto a Punto

Se basa en integración uno a uno sustentada generalmente por un middleware asincrónico basado en colas de mensajes. Si bien es un esquema de alta disponibilidad, dada por el mecanismo de comunicación, es rígido y difícil de adaptar a los cambios, además de resultar muy costoso de gestionar, monitorear y extender. Es una arquitectura accidental, completamente sincrónica, de grano grueso y poco escalable.

La Figura 21, presentada por M.Weske en (Bazán,2010), muestra el escenario de una integración con el mecanismo punto a punto.

6 Sistemas informáticos de apoyo a la gestión de las relaciones con los clientes, a la venta y al marketing (Wikipedia)
7 Es una colección de datos orientada a un determinado ámbito (empresa, organización, etc.), integrado, no volátil y variable en el tiempo, que ayuda a la toma de decisiones en la entidad en la que se utiliza (Wikipedia)

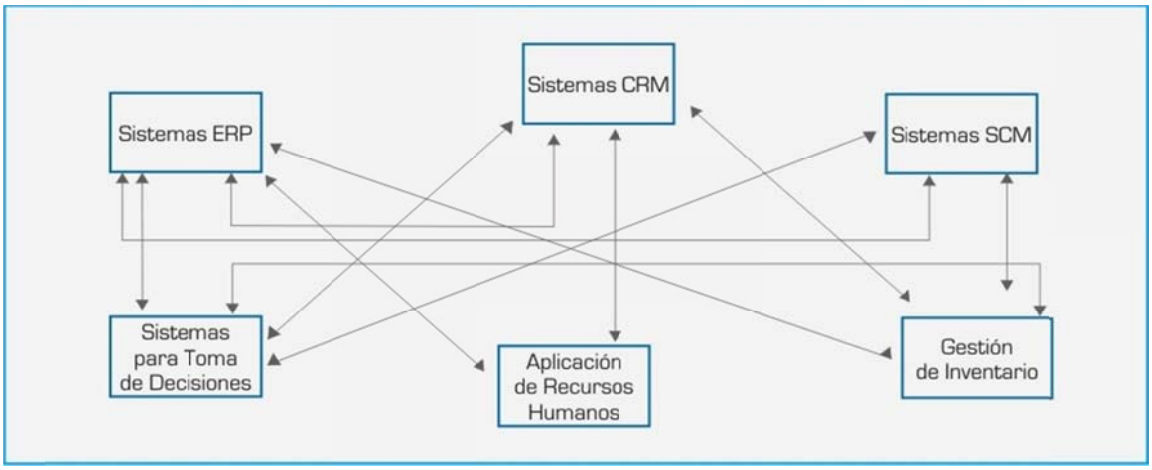

Fig. 21. Mecanismo de integración punto a punto

Integración por adaptadores

La interacción se realiza por un *hub* donde se conecta cada elemento a integrar previa construcción del adaptador correspondiente para poder *enchufarse*. Cada uno de los elementos está desconectado y no requiere utilizar el mismo lenguaje ya que existen adaptadores para establecer la comunicación.

La complejidad está en la construcción del adaptador.

La Figura 22 presenta una evolución de la Figura 21 hacia un mecanismo de integración por adaptadores.

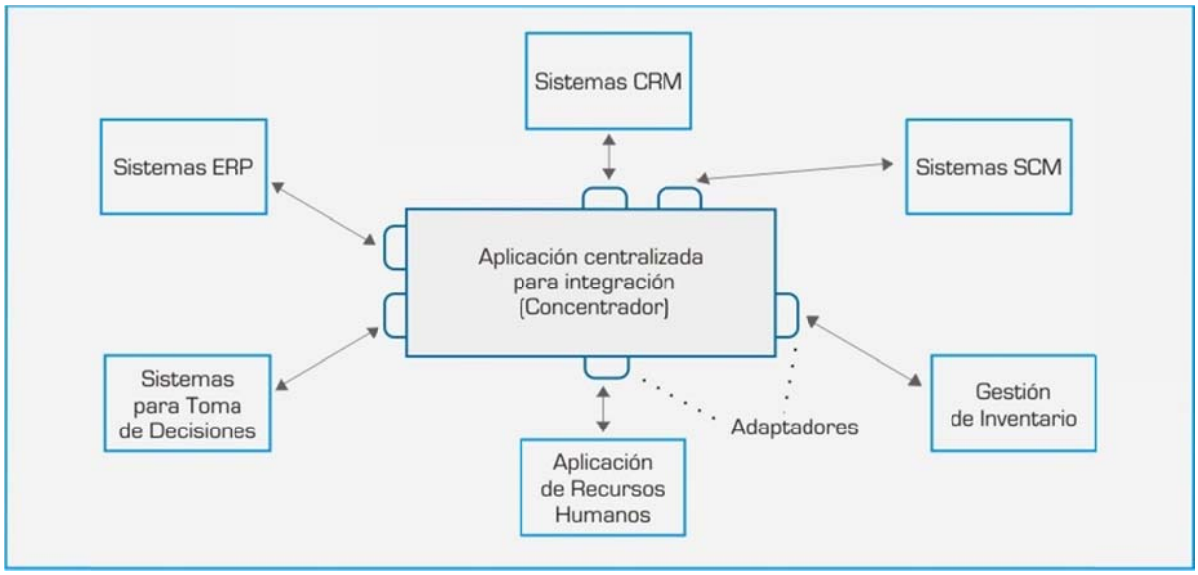

Fig. 22. Mecanismos de Integración por Adaptadores

Mediador de mensajes

Este mecanismo de integración representa una evolución del modelo anterior hacia una generalización del hub permitiendo extraer la lógica de la integración fuera de las aplicaciones. Se define declarativamente la forma de comunicación de las aplicaciones y se traslada al mediador o *broker* la lógica necesaria para producir las transformaciones que generen salidas válidas para el otro extremo.

Utiliza colas para garantizar la distribución de los mensajes, que se manejan con un esquema de publicador/suscriptor. Esto asegura que el tráfico que se genera sea solamente el requerido.

La Figura 23 muestra el próximo paso evolutivo desde la Figura 22 donde se muestra la integración a través de un broker de mensajes.

2.5. Arquitectura Orientada a Servicios

La integración de aplicaciones como se ha planteado en 2.4, si bien ha sido mejorada con diversos mecanismos como el uso de adaptadores y brokers, siguen siendo una arquitectura accidental (se construye para cada caso), de mantenimiento costoso y lento y difícil de gestionar, monitorizar y extender.

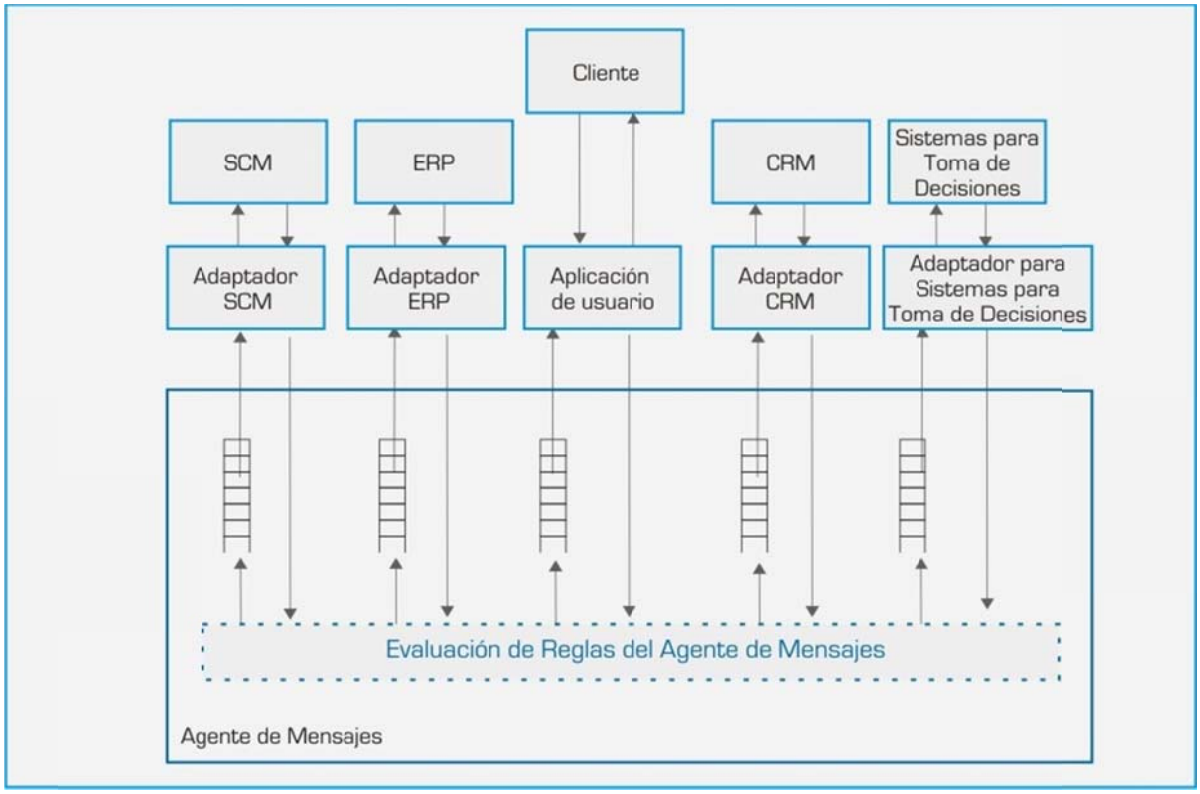

Fig. 23. Mecanismo de Integración por Mediador de Mensajes

La Arquitectura Orientada a Servicios (SOA) representa un cambio radical en la relación entre el mundo del negocio y el área de tecnología de la información. SOA constituye mucho más que un conjunto de productos aglutinados por una tecnología. Es un nuevo enfoque en la construcción de sistemas de IT que permite a las empresas aprovechar los activos existentes y abordar fácilmente los inevitables cambios en el negocio.

Si bien la industria del software ha venido enfocándose en una arquitectura orientada a servicios desde hace más de 20 años con la noción de reusabilidad y su aplicación a la construcción de software, lo cierto es que en los últimos años esto se ha fortalecido con la definición de estándares y la conformación de consorcios que participan en su definición.

Según IEEE una arquitectura de software es la organización fundamental de un sistema, reflejado por sus componentes, relaciones entre ellos y entorno, así como los principios que regirán su diseño y evolución (1471-2000) (Bazán,2010).

Según OASIS, se define como la estructura o estructuras de un sistema de información formado por entidades y sus propiedades externamente visibles, así como las relaciones entre ellas (Modelo de Referencia para SOA 1.0 – Agosto de 2006) (Bazán,2010).

Adaptándose a la definición de OASIS, se define SOA como un paradigma capaz de organizar y utilizar las capacidades distribuidas, que pueden estar bajo el control de distintas organizaciones, y de proveer un medio uniforme para publicar, descubrir, interactuar y usar los mecanismos oportunos para lograr los efectos deseados.

Podemos resumir los conceptos subyacentes fundamentales en este paradigma en los siguientes:

- Proveedor: entidad (organización o persona) que ofrece el uso de capacidades mediante servicios.
- Necesidad: carencia de una empresa para resolver la actividad de su negocio.
- Consumidor: entidad (organización o persona) que busca satisfacer una necesidad particular a través de las capacidades ofrecidas por servicios.
- Capacidad: tarea que el proveedor de un servicio puede proporcionar al consumidor.
- Servicio: mecanismo que permite el acceso a una o más capacidades alcanzables por medio de una interfaz preestablecida, y que se llevará a cabo de forma consistente con las normas establecidas para él.
- Descripción del servicio: información necesaria para hacer uso del servicio. Interacción: actividad necesaria para hacer uso de una capacidad con el objeto de obtener efectos deseados.

En el Capítulo 5 se retoman y amplían los conceptos de Arquitecturas Orientadas a Servicios.

2.6. Procesos de Negocio como Consumidores de Servicios

En términos generales, cuando se habla de integración de aplicaciones nos referimos tanto a datos como a procesos.

Los datos pueden integrarse o puede resolverse su integración contando con cualquiera de los esquemas anteriormente planteados. Cualquiera de ellos implica el desencadenamiento de un flujo o cadena de mensajes que quedan incrustados dentro del esquema de integración mismo.

Así, los sistemas de gestión de workflow surgen como ciudadanos de primera clase a la hora de realizar una integración de aplicaciones eficiente y fácil de mantener y actualizar.

Gran parte del esfuerzo de implementar una arquitectura orientada a servicios orquestada por procesos de negocio, se ha centrado sobre la integración de aplicaciones y el workflow. Sin embargo existe una preocupación de los arquitectos de software acerca de la incapacidad para acceder a datos de negocios y administrarlos de una manera ágil tal como sucede con la lógica de negocio en las aplicaciones.

La integración de datos dirigida por procesos puede ayudar a enriquecer los servicios de negocios SOA y los procesos de negocio a través de una secuencia de servicios de datos combinados de manera reusable que incorpora la intervención de tareas humanas transformando la información en exacta, consistente y oportuna.

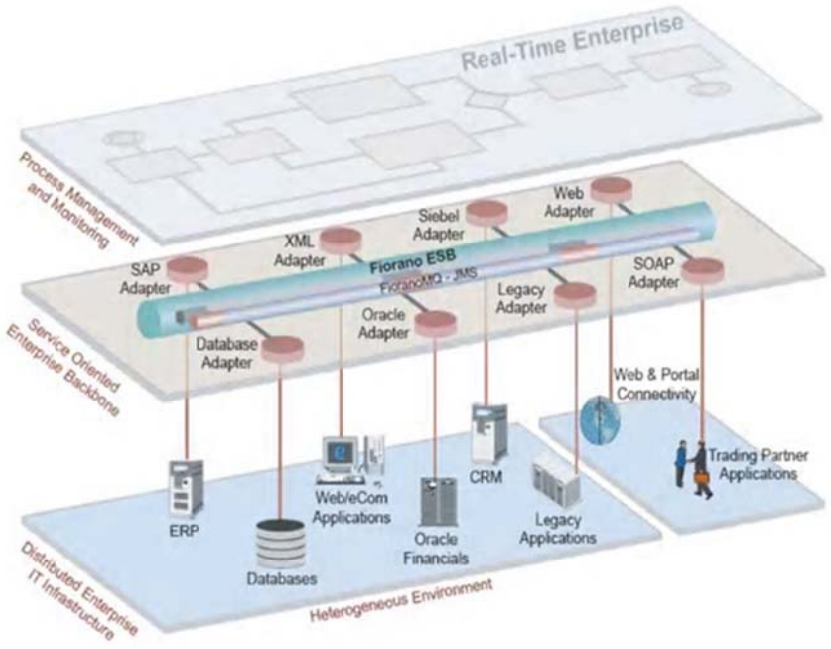

Fig. 24. Procesos de Negocio como Consumidores de Servicios

2.7 Conclusiones del Capítulo

Este capítulo presenta la evolución tanto cronológica como en complejidad de los sistemas distribuidos. El esquema de distribución planteado se corresponde tanto con lo funcional como con lo estructural, siendo la combinación de ambos tipos de distribución lo que hace más complejas las soluciones.