

# Construcción de un Framework para la Definición, Acceso y Consulta de Fuentes de Información Heterogéneas\*

Hernando Bustos      Victor Chambe  
hbustos@uta.cl      Vchambe@123mail.cl  
Departamento de Comp. e Informática  
Universidad de Tarapacá

## Resumen

Los cambios que ocasiona un mundo globalizado, como el actual, exigen a los sistemas de información una adaptación rápida para que satisfagan las necesidades de información que las organizaciones requieren para enfrentar este nuevo escenario.

Sin embargo, la mayoría de los proyectos de adaptación de sistemas de información, se ven enfrentados a un proceso de estructuración difícil y prolongado. La experiencia, indica que el mayor porcentaje de tiempo y preocupación se consume en la integración de los distintos tipos de Sistemas de Información de Empresas (SIEs), construyendo soluciones particulares, mediocres y dependientes, que en muchos casos no pueden ser utilizadas en futuros desarrollos.

Hoy en día, aplicando tecnologías de reuso, es posible poder pensar en disponer de un conjunto de componentes de software reusables (framework) que provean facilidades para la integración de SIEs, permitiendo a los desarrolladores concentrarse mayoritariamente en las soluciones requeridas, en vez de la integración. El presente documento, describe el trabajo que se está realizando para el desarrollo de un framework de este tipo.

**Palabras Claves:** Framework, Componentes, Reuso, Sistemas de Información de Empresas.

## 1. Introducción.

La tendencia a un mundo globalizado ha cambiado la economía y la manera de hacer negocios. En este contexto la gestión de la información es fundamental en el éxito o fracaso de una organización. Dado que el acceso e intercambio de información es común en una economía basada en el conocimiento, crear métodos eficaces para llevar a cabo esta tarea es primordial.

Un Sistema de Información de Empresa (en adelante SIE) se define como un sistema de hardware y software que proporciona la infraestructura de información que una empresa usa para desarrollar sus negocios, y que mediante una interfaz ofrece un conjunto de servicios para recuperar y actualizar la información que mantiene. Ejemplos de SIE son los Sistemas Gestores de Bases de Datos (Oracle, SQL Server, Sybase, DB/2), Sistemas Mainframe de Procesamiento de Transacción (CICS), Sistemas de Planificación de Recursos de Empresa (FlexLine, SAP, CRM), Sistemas Propietarios de almacenamiento, Sistemas legados, etc.

Los cambios que ocasiona este proceso de globalización (ampliar sus fronteras, reducir sus costos y tiempos de respuesta, proporcionando continuamente nuevos servicios de información de fácil acceso a sus clientes, socios, empleados y proveedores), exigen a los sistemas de información una adaptación rápida para que satisfagan las necesidades de información que las organizaciones requieren para enfrentar este nuevo escenario.

Sin embargo, la mayoría de los proyectos de adaptación de sistemas de información que se desarrollan por efecto de necesidades de formación de alianzas, realización de fusiones, contar con servicios de información de otras empresas o necesidades de evolución de sistemas para mantener la competitividad; se ven enfrentados a un proceso de estructuración difícil y prolongado, debido a la existencia de problemas, tales como:

---

\* Financiado por proyecto Fondecyt 2990070 y proyecto Mayor UTA.871201

- La información que los nuevos sistemas de información deben recopilar, consolidar y presentar está repartida en un número cada vez mayor de SIEs.
- Las organizaciones, desean mayoritariamente mantener sus antiguos SIEs, ya sea por la alta inversión realizada o por el costo que significaría una migración.
- Desarrollo de nuevas aplicaciones acorde a las nuevas tecnologías que sean capaces de interactuar con sistemas legados, destinadas a cubrir necesidades de clientes, usuarios y empresa no satisfechas.

La experiencia en el desarrollo de este tipo de proyectos de aplicaciones, muy comunes y recurrentes hoy en día, indica que es un problema de alta complejidad. Donde el mayor porcentaje de tiempo se consume en la integración de los distintos tipos de SIEs, construyendo soluciones particulares para acceder cada uno de ellos, lo que deriva en soluciones específicas, mediocres y dependientes, que en muchos casos no pueden ser utilizadas en futuros desarrollos. Si a esto se le agrega la necesidad de cortos tiempos para el desarrollo y las crecientes exigencias de calidad del software construido, se termina por configurar un problema de marca mayor.

Hoy en día, aplicando tecnologías de reuso, es posible poder pensar en disponer de un conjunto de componentes de software reusables que provean facilidades para la integración de SIEs, permitiendo a los desarrolladores concentrarse en las soluciones requeridas, en vez de la integración. Esta, es una alternativa coherente con las necesidades de desarrollo en corto tiempo y con un alto nivel de satisfacción de los clientes y usuarios.

En el contexto de orientación a objetos, un conjunto de componentes reusables en un dominio específico, como el anterior, se denomina *framework* [Froe97, John97]. Un framework provee a los desarrolladores soluciones para una familia de problemas y mantener mejor estas soluciones. Proporciona un buen diseño y una infraestructura pensada para que sus componentes puedan ser reemplazadas con el menor impacto sobre otras componentes del framework.

Taligent [Tali94] define que los frameworks pueden ser agrupados, según las funcionalidades que abarcan, en tres categorías: *Framework de aplicación*, *Framework de dominio*, *Framework de soporte*.

En lo que resta del documento, se describe el trabajo que se está realizando para el desarrollo de un framework de aplicación orientado a proveer la integración de SIEs. Para ello, en la sección 2 se detalla la definición de objetivos y requerimientos del framework, en la sección 3 se presenta la arquitectura que tendrá el framework, en la sección 4 se detalla el avance en el diseño, en la sección 5 se describe la plataforma de implementación a usar, y en la sección 6 se plantea algunas conclusiones temporales.

## 2. Definición del Framework

Los frameworks vistos como arquitecturas reusables están estrechamente ligados a otra aproximación para el reuso como son los *Patrones*. Un patrón para una arquitectura de software describe un problema recurrente y particular de diseño que se presenta en contextos específicos, y entregan un esquema genérico bien probado para su solución [Busc96]. Ellos Capturan la intención detrás de un diseño identificando objetos, sus colaboraciones, y la distribución de responsabilidades [Gamm93]. Los patrones pueden ser vistos como bellos ejemplos de buen diseño, que resuelven problemas recurrentes, aplicados y probados exitosamente en el mundo real, que plasman la experiencia de gente experta, y que pueden ser reusados en muchos dominios.

Luego entonces, no es coincidencia la similitud que existe entre los conceptos de framework y patrones, pues ambos persiguen el reuso. Los frameworks pueden ser vistos como una construcción concreta de un conjunto de familias de patrones de diseño que apuntan a un dominio particular de aplicación y los patrones de diseño pueden ser vistos más como elementos microarquitecturales abstractos del framework que documentan y motivan las semánticas de frameworks de una manera efectiva [Faya97]. De esta manera, basándonos en patrones, podemos asegurar la robustez, extensibilidad y perdurabilidad de nuestra solución.

Dado lo anterior, se propone como alternativa de solución al problema de desarrollo de aplicaciones que requieren interactuar con diversos SIEs, la construcción de un framework de aplicación usando patrones de diseño,

que permita mediante un método común dar soporte para la definición, acceso y consulta de información de una variedad de tipos de SIEs.

En la actualidad existe en el mercado software de integración comercial, tales como, Oracle 9iAS Integration, WebSphere MQ Integrator o DCOM. Estos productos denominados middleware permiten integrar un conjunto de SIEs y proporcionan servicios de ruteo y encolamiento de mensajes, transformaciones, y construcción de servicios web, entre otros. Sin embargo, es posible apreciar en ellos algunas desventajas tales como.

- **Costo:** Ante todo, son "comerciales", por lo que en algunos casos es necesario desembolsar sumas considerables de dinero. Adicionalmente, este carácter comercial, lleva a la existencia de un **Código Cerrado**.
- **Portabilidad:** No son del todo portables, en el caso de Oracle 9iAS Integration y WebSphere MQ Integrator corren respectivamente en los servidores Oracle 9iAS y WebSphere, los cuales, a pesar de ser servidores J2EE "compliant", no es posible instalar Oracle 9iAS Integration en WebSphere ni en cualquier otro servidor J2EE, lo mismo ocurre para WebSphere MQ Integrator. En el caso DCOM, es bien conocido que corre en Windows, y sólo en Windows.
- **Uso de estándares:** No consideran estándares en todos los niveles de sus soluciones (solo a nivel de Servicios Web: uso de estándares como SOAP, HTTP, FTP, SMTP, UDDI, WSDL). Oracle 9iAS no soporta JCA (Java Connector Architecture), sin embargo, expresa que en el futuro lo soportará.
- **Integración:** Oracle 9iAS InterConnect y WebSphere MQ Integrator no consideran integración con productos de la competencia (otros SGDB), salvo que su uso este muy extendido.

## **2.1. Objetivos**

Para el proyecto a realizar se plantea el siguiente objetivo general: “Construir un framework de aplicación utilizando patrones de diseño que provea a los desarrolladores de sistemas de información un método común para definir, acceder y consultar un conjunto heterogéneo de fuentes de información”.

Para el logro de esto, se plantea los siguientes objetivos específicos.

- Definir la Arquitectura del Framework.
- Construir usando patrones de diseño, los componentes que permitan dar soporte a los distintos tipos de SIEs.
- Definir y construir los mecanismos para configurar el framework.

## **2.2. Requerimientos.**

Dado que el framework debe resolver el problema de acceso e interacción con SIEs con fuentes de información heterogéneas de una manera común, se define los siguientes requerimientos funcionales.

Proveer un servicio de interfaz transaccional con las aplicaciones.

Proveer mecanismos de parametrización, por medio de la definición de SIEs y transacciones disponibles para las aplicaciones.

Proveer mecanismos de extensibilidad, por medio de la definición de nuevos tipos de SIEs y transacciones.

Adicionalmente se considera los siguientes requerimientos no funcionales.

- Construido usando patrones.
- Debe ser portable.
- Debe ser independiente de los SIEs subyacentes.

Como consecuencia, se espera obtener una serie de ventajas, tales como:

- Disponibilidad de una base de desarrollo de aplicaciones que requieren tener acceso a un conjunto de tipos de SIEs testeadas y aprobadas.
- Disminución de los tiempos y costos de desarrollo de sistemas de información que requieran la integración de SIEs.
- Bajo costo de mantención de los servicios de datos.
- Bajo acoplamiento entre los servicios de datos y los tipos de SIEs a los que están asociados.
- Aumento de la calidad de los productos obtenidos.

### 3. Arquitectura del Framework.

Se visualiza para este framework una estructura como la que se muestra en la Figura 1. En ella se puede apreciar que para el desarrollo de una aplicación será necesario instalar en el Servidor de aplicaciones el Servicio Transaccional, el Soporte para los tipos de SIEs con los que la aplicación interactuará, la Definición de los tipos de SIEs y transacciones soportados, y por último la Definición de SIEs accesados y transacciones disponibles para la aplicación. A continuación, se describe cada uno de estos componentes.

- Servidor de aplicaciones: Servidor que hospedará todas las componentes aquí descritas.
- Framework (Servicio Transaccional): Implementa el Servicio Transaccional y proporciona las interfaces e implementaciones base para que proveedores de soporte desarrollen extensiones del framework que permitan al Servicio Transaccional soportar nuevos tipos de SIE.
- Soporte para tipo de SIE n: Conjunto de componentes que dan soporte para un tipo de SIE. El proveedor de soporte construye, en base a las interfaces e implementaciones base del framework, un conjunto de componentes que despliega en el *Servidor de aplicaciones* mediante los archivos de definición asociados al *Servicio Transaccional*.
- Definición de tipos de SIE y Transacciones soportados: Describe los tipos de SIE y transacciones soportados.
- Definición de SIEs y Transacciones: SIEs y Transacciones asociadas a los que el *Servicio Transaccional* tiene acceso, disponibles para la aplicación.
- Aplicación: Sistema que usa las transacciones instaladas y disponibles mediante el *Servicio Transaccional*.

En el caso de uso principal, la aplicación solicitará al Servicio Transaccional la transacción que requiere ejecutar. Para ello el Servicio Transaccional, para cada transacción, revisa en la Definición de SIEs y Transacciones que exista una definición para dicha transacción y un SIE a la que esté asociada. Luego, basándose en la definición de tipos de SIEs y Transacciones soportados, determina el conjunto de componentes (que implementan las interfaces que provee el Framework) que permiten la instanciación, ejecución y generación del resultado de la transacción. Con la información reunida, la transacción es instanciada e iniciada y por último es retornada a la aplicación para su posterior iniciación de parámetros y ejecución.

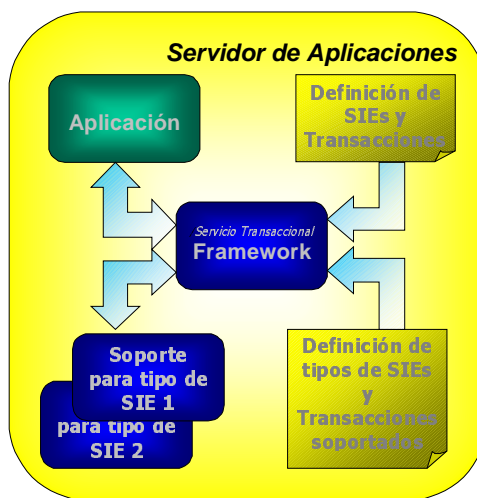


Figura 1 Arquitectura del Framework

### 4. Diseño del Framework.

Se mostrará el diseño del framework a través del flujo normal de operación que este tendrá en la interacción con las aplicaciones, desde el inicio hasta el término del servicio transaccional.

#### 4.1. Inicio del Servicio Transaccional.)

En la Figura 2 se puede observar que cuando el Servicio Transaccional(TransactionalService) sea iniciado por el Administrador del Servicio Transaccional (TransactionalServiceManager) cargará las definiciones de tipos de SIEs (EISTypeDescriptor) y transacciones (TransactionTypeDescriptor) soportados y las definiciones de SIEs accesados (EISDescriptor) y sus transacciones asociadas (TransactionDescriptor), quedando a la espera de los requerimientos de la Aplicación.

#### 4.2. Obtención de transacciones mediante el Servicio Transaccional.

De igual manera en la Figura 2 se observa que para cada transacción solicitada al Servicio Transaccional, este, revisa en la Definición de SIEs y Transacciones que exista una definición para dicha transacción y un SIE a la que esté asociada (en caso contrario lanza una excepción –TransactionalServiceException). Luego, basándose en la definición de tipos de SIEs y Transacciones soportados, determina el conjunto de componentes (que implementan las interfaces que provee el Framework) que permiten la instanciación, ejecución y generación del resultado de la transacción. Con la información reunida genera una instancia de especificación de transacción (TransactionSpec), la transacción (Transaction) es instanciada, iniciada (ConnectionFactory, ResultFactory, Parameters) y retornada a la aplicación. Cada tarea que realiza el Servidor Transaccional es registrado en una bitácora (Log).

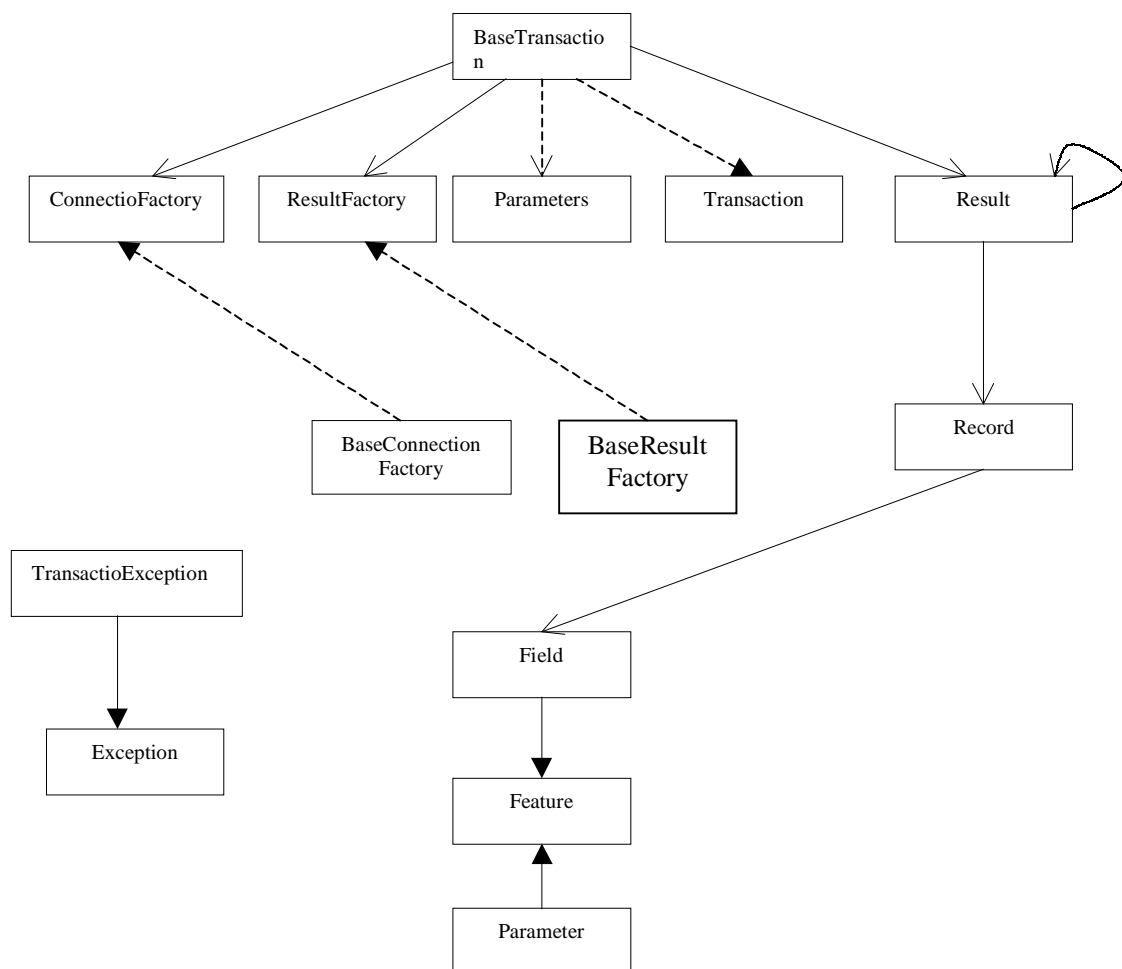


Figura 2. Interfaces e Implementaciones Base

### 4.3. Iniciación y ejecución de transacciones.

En la Figura 3, se puede observar que una vez que la Aplicación ha obtenido la transacción mediante el Servicio Transaccional, inicia sus parámetros y procede a su ejecución. La implementación base (BaseTransaction) de una transacción (Transaction) incorpora un esquema genérico para la ejecución de transacciones que debe ser implementada por el tipo de transacción específico (Por ej. JDBCStatementTransaction), que consta de los siguientes pasos:

- Inicia el resultado de la transacción a nulo.
- Solicita a la Fábrica de conexiones (ConnectionFactory), asociado al tipo de SEI (Por ej. JDBCConnectionFactory), una conexión al SEI.
- Construye una instrucción que permita la ejecución de la transacción en el SEI.
- Inicia los parámetros de la instrucción (Parameters). Un parámetro( Parameter) es una característica (Feature) que tiene nombre, tipo y valor.
- Ejecuta la instrucción.
- Obtiene el resultado genérico de la transacción, mediante la Fábrica de resultados (ResultFactory) asociada al tipo de SEI (Por ej. JDBCResultFactory), y lo asigna como resultado de la transacción. Un Resultado (Result) estará compuesto de un conjunto de registros (Record), y un registro de un conjunto de campos (Field), y un campo se define como una característica (Feature).
- Retorna la conexión a la Fábrica de conexiones.

Si en algún punto existe un problema, la transacción lanza una excepción (TransactionException). Ahora la Aplicación ya esta en condiciones de manipular el resultado de la transacción.

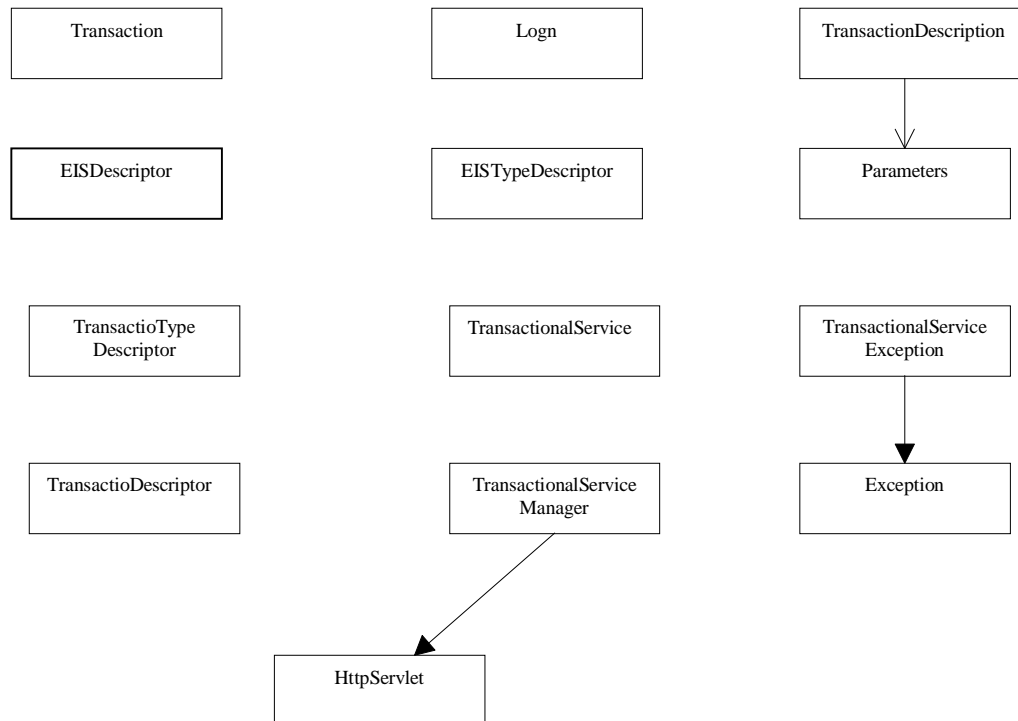


Figura 3. Servicio Transaccional.

#### **4.4. Término del Servicio Transaccional.**

Nuevamente, en la Figura 2 se puede observar que cuando el Servicio Transaccional (TransactionalService) sea terminado por el Administrador del Servicio Transaccional (TransactionalServiceManager), este liberará los recursos y no aceptará más requerimientos de la aplicación.

### **5. Plataforma de Desarrollo.**

La plataforma en que se está comenzando a desarrollar este framework es J2EE. Los principales factores que influyeron en su selección fueron:

- J2EE define el estándar para el desarrollo de aplicaciones empresariales multicapa.
- Constituye una plataforma sumamente robusta y en continua evolución debido a que está basado en un conjunto de especificaciones (y no implementaciones como .NET de Microsoft) de tecnologías bien definidas en las que han participado importantes empresas, tales como, Oracle, IBM, Sun, Sybase y BEA Systems.
- Incorpora estándares ampliamente aceptados de grupos como W3C y XOPEN.
- Recibe la cooperación diversos proyectos open source.
- El modelo de aplicación J2EE provee una aproximación simplificada para el desarrollo de aplicaciones distribuidas, altamente escalables y tolerante a fallas, basadas en intranet o internet.
- J2EE toma ventaja de muchas características de J2SE, tales como: portabilidad mediante el concepto “Write Once, Run Anywhere” (escriba una vez, córralo donde quiera), la API JDBC para el acceso de bases de datos, la tecnología CORBA para la interacción con recursos de empresas existentes y un modelo de seguridad que protege los datos incluso en aplicaciones internet.
- Hoy en día, los principales vendedores de Servidores de Aplicación, SEIs y herramientas de desarrollo, han adoptado e introducido productos basados en la especificación de la plataforma J2EE, ya que el uso de un único estándar, bajo un único lenguaje, asegura la compatibilidad e interoperabilidad entre implementaciones y plataformas de distinto origen, que estimula obviamente una sana competencia.
- Por último, la disponibilidad.

### **6. Comentarios y Conclusiones.**

- El Servicio Transaccional está habilitado para ser usado en aplicaciones que corren en el mismo Servidor de Aplicaciones. Esto representa una restricción importante para el tipo de aplicaciones que se construyen hoy en día. Una aplicación moderna publicada en internet o intranet son típicamente distribuidas.
- La revisión de las alternativas y la visión que se tiene con respecto al futuro nos lleva a pensar en la implementación de un Servicio Transaccional Web como solución idónea.
- Tal como se concibe permite integrar cualquier SGBD que posea un driver JDBC u ODBC ocultar al desarrollador de aplicaciones la complejidad de las APIs específicas para la integración de SIEs y reemplazarlas por APIs sumamente sencillas.
- Este Servicio Transaccional Web se basaría sobre el concepto de Servicios Web(Web Service, W3C) que permite a aplicaciones internet o intranet proveer y publicar servicios en un lenguaje independiente de la plataforma como es XML (eXtensible Markup Language) para que sean usados por otras aplicaciones (esto es especialmente atractivo en aplicaciones B2B).
- Esto último sería totalmente coherente con el concepto de independencia de los tipos de SIE que persigue el Framework de Servicio Transaccional.

## 7. Bibliografia.

[Busc96] Buschmann, Frank, and Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. Pattern-Oriented Software Architecture: A system of Patterns. England: Wiley, 1996.

[Faya97] F. Mohamed, D.C.Schmidt. Object-Oriented Application Frameworks, Communications of the ACM, Special Issue on Oriented Application Frameworks, Vol.40, No.10, 1997.

[Fowl97] Fowler, Martin. Analysis Patterns: Reusable Object Models. Massachussets: Addison Wesley, 1997.

[Froe97] G. Froehlich , J. Hoover , L. Liu , and P. Sorenson . Hooking into object-oriented application frameworks. Software engineering , 1997, 491 – 501.

[Gamm93] E.Gamma, and R.Helm, R.Johnson, J. Vlissides. Design Patterns: Abstraction and Reuse of Object-Oriented Design, ECOOP '93 Conference Proceedings, Springer-Verlag Lecture Notes in Computer Science.

[Gamm95] E. Gamma, R. Helm, R. Johnson and J. Vlissides. Design Pattern: Elements of Reusable Object Oriented Software. Addison Wesley, 1995.

[Hans97] T. Hansen, S. Fraser, C. Hilsenrath, B. Opdyke, A. Riel. Development of Succesful Object-oriented Frameworks. OOPSLA'97 Frameworks Workshop.

[Tali94] Taligent, Inc., “Building Object-Oriented Frameworks”,  
<ftp://www6.software.ibm.com/software/developer/library/buildingoo.pdf>

[John97] R. E.Johnson. Frameworks = (components + patterns).Comm. ACM 40, 10 (Oct. 1997), 39 – 42.