UNIVERSIDAD PONTIFICIA DE SALAMANCA

Campus de Madrid

Facultad de Informática

DOCTORADO EN INGENIERÍA INFORMÁTICA

Programa en Ingeniería del Software

BIENIO 2003-2005

ASIGNATURA:

Diseño Avanzado de Sistemas Operativos

TRABAJO DE INVESTIGACIÓN:

Análisis de Mejora de flexibilidad de la Arquitectura de Desarrollo de Aplicaciones Bancarias "Banksphere" mediante la aplicación de técnicas de Reflectividad.

PROFESOR:

Dr. D. Darío Álvarez Gutiérrez

AUTOR:

César Parejas Llanovarced

Madrid, Febrero de 2005

ÍNDICE

INTRODUCCIÓN

Se dice que una arquitectura de software es flexible cuando cuenta con la capacidad de diseñar sistemas que puedan ser ajustados a las necesidades de aplicaciones o dominios de aplicación específicos. En este sentido, la demanda de las nuevas aplicaciones de software, sistemas distribuidos, la creciente demanda del uso de ordenadores en nuevas áreas, etc. requieren cada vez más del soporte de arquitecturas flexibles.

La capacidad de reflexión de una arquitectura (reflectividad) posibilita el análisis y modificación de él mismo. Existen diversas líneas de investigación entorno a la reflectividad así como sistemas comerciales que hacen uso, en mayor o menor grado, de ella. La reflectividad computacional es una clasificación del concepto de reflectividad por la cual se puede modificar la semántica o interpretación (el funcionamiento) de un sistema. Es en este punto donde existen más líneas de investigación abiertas y donde todavía no se ha empleado de forma clara en sistemas comerciales.

La flexibilidad en la especificación de una plataforma independiente hace de la reflectividad un criterio interesante para el diseño de nuevas arquitecturas.

El presente trabajo de investigación tiene por objetivo analizar las características de una arquitectura de desarrollo actual (Banksphere) basada en un modelo de tres capas y con tecnología J2EE, presentando sus ventajas y desventajas. A partir de dicho análisis, planteamos la posibilidad de incrementar sus características de flexibilidad mediante la aplicación de técnicas de Reflectividad Computacional y SIOO (sistema integral orientado a objetos).

SISTEMA INTEGRADO PARA EL DESARROLLO DE APLICACIONES MULTICANAL

Qué es Banksphere

Banksphere es un entorno completo de desarrollo y ejecución de aplicaciones distribuidas para el mundo Web, basado en J2EE (Java 2 Enterprise Edition). Sigue un modelo de tres niveles que diferencia las capas de presentación, de lógica de negocio y de modelo de datos.

Banksphere proporciona un conjunto de herramientas para el diseño, creación y mantenimiento de los diferentes componentes, permitiendo la gestión de los mismos desde su concepción hasta su puesta en producción. La plataforma de integración de las herramientas de desarrollo de Banksphere es WSAD (WebSphere Application Developer).

Estándares

La metodología está basada en el Rational Unified Process (RUP) y emplea el estándar Unified Modeling Language (UML) como forma de expresar y documentar el comportamiento y funcionalidad de las aplicaciones.

El entorno de ejecución de Banksphere está basado en la plataforma Java 2 Enterprise Edition J2EE 1.3 y posteriores, y como, tal adopta todos los estándares definidos para esta plataforma. En particular, entre otros:

- Servlet Specification level of 2.3.
- JSP Specification level of 1.2.
- Enterprise JavaBeans Specification 2.0.

El empleo de estándares de mercado, es un principio de diseño que marca la estrategia de desarrollo y construcción de Banksphere.

Arquitectura

Entorno de desarrollo

El entorno de desarrollo está fundamentado en el estándar Eclipse 2.1. Todas las herramientas empleadas son extensiones, en forma de conectores, a esta plataforma de desarrollo.

Al tratarse de un entorno J2EE (Java 2 Enterprise Edition) el lenguaje de desarrollo por excelencia es Java. Para la construcción de la parte visual se

emplean los lenguajes clásicos utilizados en los navegadores, HTML y JavaScript, sin estar cerrado a la utilización de cualquier recurso estándar.

Los lenguajes de desarrollo empleados son:

- Para la construcción de componentes de negocio, tipo EJB (Enterprise Java Beans), se emplea Java.
- Para la construcción de los componentes visuales se emplea JavaScript y HTML.

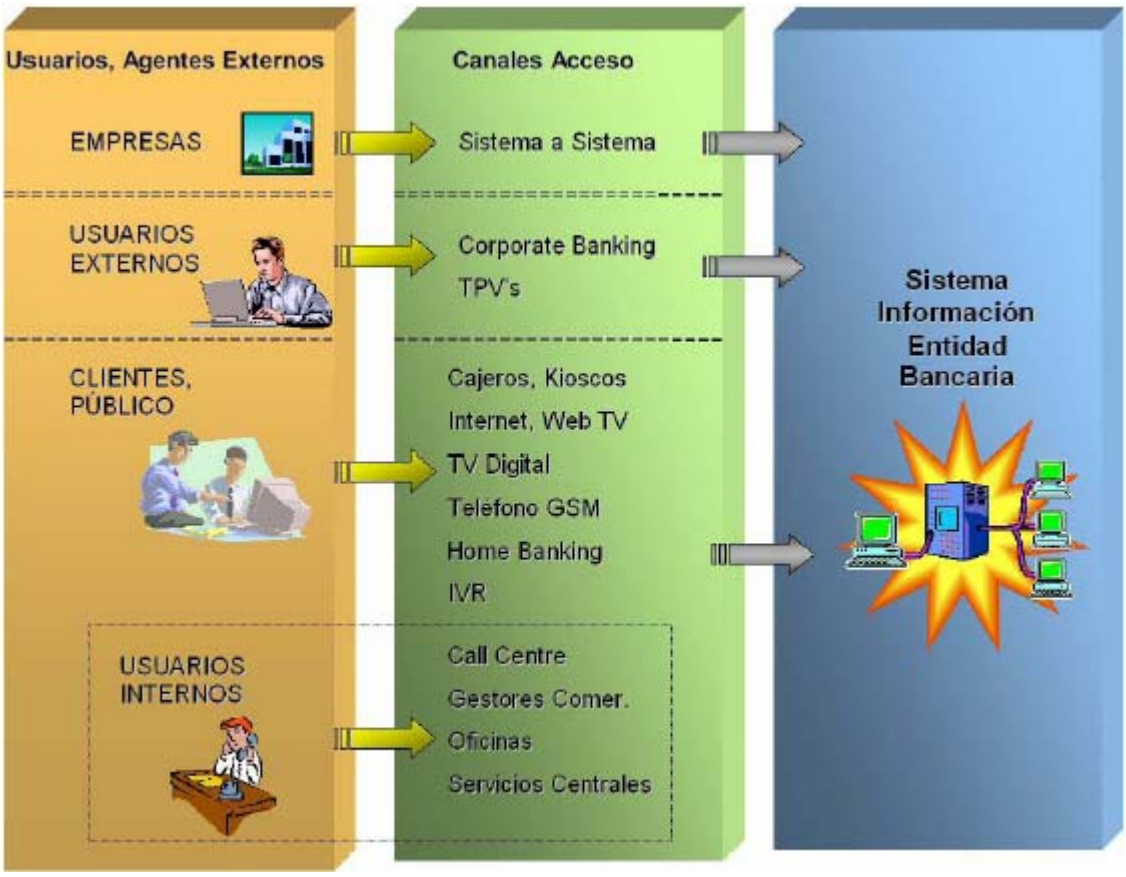

Figura 1. Ciclo de desarrollo

Arquitectura de ejecución

El run-time de Banksphere es un motor de aplicaciones gestionado y gestionable. Facilita que el rendimiento y comportamiento de las aplicaciones desplegadas sobre él sea predecible.

Permite que el desarrollador se concentre en resolver el problema de negocio y se despreocupe de los mecanismos técnicos de la plataforma.

Características del entorno de ejecución:

- Arquitectura de 3 capas, basada en los siguientes componentes:

- Componentes clientes.
- Componentes Web.
- Componentes de negocio.

- Es multicanal, multidioma y multientidad.
- Basada en el cliente ligero, esto es, utiliza una interfaz de tipo *browser* empleada para acceder a la aplicación.
- Incrementa la productividad, la reutilización y la conformidad con los estándares.

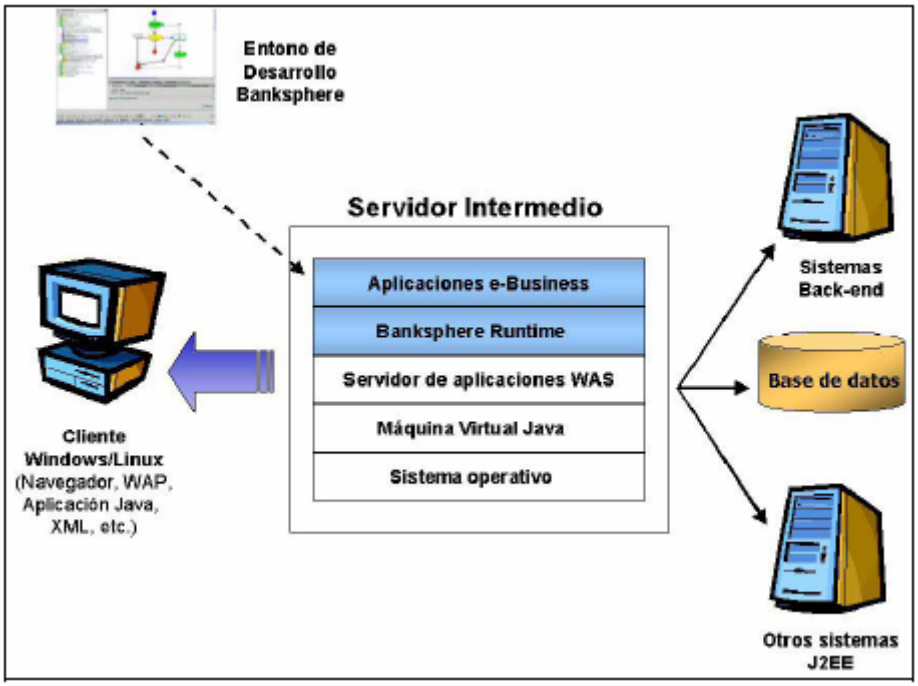

Figura 3. Entorno de ejecución

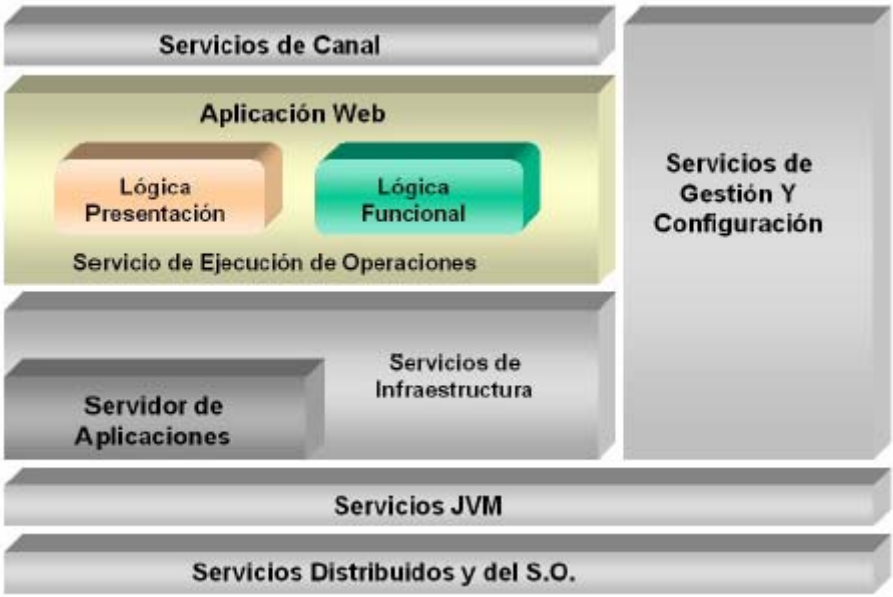

Figura 4. Visión por capas

VENTAJAS DE LA ARQUITECTURA BANKSPHERE

- **Escalabilidad y rendimiento**
  Banksphere permite aumentar la respuesta de las aplicaciones distribuidas en función de la carga. Las aplicaciones construidas sobre Banksphere permiten soportar desde pocas a muchas peticiones sin modificación alguna en el código.

- **Alta disponibilidad y balanceo de carga**
  Banksphere, apoyada en los mecanismos proporcionados por los servidores de aplicaciones, permite que varios procesadores cooperen a la hora de ejecutar operaciones. Además de repartir la carga entre los procesadores de un cluster, permite que, en caso de producirse la caída de uno de ellos, los demás asuman la carga y la aplicación siga estando disponible.

- **Seguridad**
  Banksphere incluye un servicio de autenticación de usuarios que facilita la gestión del acceso de estos a las aplicaciones mediante la utilización de una credencial de usuario.

DESVENTAJAS DE LA ARQUITECTURA BANKSPHERE

El principal problema que se presenta en el desarrollo de aplicaciones bajo esta arquitectura, es que el paradigma de orientación a objetos no es adoptado de una forma integral por todos los componentes del sistema que lo constituyen. Esto trae como consecuencia:

- **Desadaptación de impedancias**
  Puesto que existe un salto semántico ocasionado cada vez que un componente del sistema (ej. sistema legacy o sistema operativo) debe interactuar con otro (ej. sistema orientado a objetos).

- **Problemas de Interoperabilidad entre modelos de aplicaciones**
  Asi, una aplicación desarrollada usando el lenguaje Java, con el modelo de objetos Java, no tiene ningún problema de comunicación con sus propios objetos, pero cuando se desea interactuar con datos o componentes de otras aplicaciones, como una aplicación desarrollada en el Leguaje COBOL, con modelo de datos relacional en DB2 (que representan un alto porcentaje en la automatización de funcionalidad bancaria), aparece un problema de interoperabilidad. Esto debido a que el modelo de objetos Java no tiene por qué ser compatible con el de otros componentes.

Para solucionar estos problemas, la arquitectura Banksphere, como cualquier otra plataforma de desarrollo convencional, recurren a la introducción de capas de software de adaptación adicionales, lo que ocasiona una disminución del rendimiento global del sistema, una pérdida de portabilidad y flexibilidad y un aumento de la complejidad.

PROPUESTA DE INCREMENTO DE FLEXIBILIDAD A LA ARQUITECTURA BANKSPHERE

Sistema integral orientado a objetos

Como alternativa de solución a este problema, se plantea la construcción de un sistema integral orientado a objetos, en donde todos sus componentes: interfaces de usuario, aplicaciones, lenguajes, compiladores, bases de datos y el propio sistema operativo compartan el mismo paradigma de orientación a objetos.

Una técnica para estructurar un sistema operativo orientado a objetos que soporte un sistema integral orientado a objetos es emplear una máquina abstracta como base del sistema operativo.
El sistema de desarrollo integral orientado a objetos, al ser un sistema construido sobre un sistema operativo integrado a objetos (empleando por ejemplo una máquina abstracta como base del sistema operativo), compartirá el mismo paradigma de orientación a objetos (desapareciendo la desadaptación de impedancias). Además el propio sistema operativo proporcionaría una serie de características tales como persistencia, concurrencia, distribución y seguridad, de las que Banksphere se aprovecharía.

Con este planteamiento, lo que se pretende es conseguir:

- **Una arquitectura de desarrollo más sencilla.** Asumiendo que algunas de las funciones que debería contemplar el desarrollador (ej. Persistencia, Seguridad) ya están disponibles dentro del propio sistema operativo.
- **Mayor rendimiento.** Dado que el propio sistema integral ya es orientado a objetos, no existe la necesidad de desarrollar capas superpuestas como en una arquitectura tradicional para salvar el espacio existente entre el paradigma del sistema operativo y las aplicaciones bancarias.
- **Mayor productividad.** La programación de aplicaciones bancarias es más productiva ya que no es necesario que el programador cambie constantemente de filosofía de trabajo.

Reflectividad Computacional

Visión general

Reflectividad o reflexión (reflection) es la propiedad de un sistema computacional que le permite razonar y actuar sobre él mismo, así como ajustar su comportamiento en función de ciertas condiciones.

Un sistema reflectivo es un sistema que incorpora una estructura que lo representa (o aspectos de él). Llamaremos a la suma de estas estructuras la auto representación del propio sistema. Ésta auto representación hace posible al sistema responder a cuestiones acerca de si mismo y dar soporte a sus propias acciones. Puesto que la auto-representación es conexa directamente a los aspectos del sistema que representa, se puede decir que:

- El sistema siempre tiene una representación exacta de si mismo.
- El estado y comportamiento de los sistemas están siempre en conformidad con su representación. Esto significa que un sistema reflectivo puede de hecho acarrear modificaciones en el propio sistema como consecuencia de su propia computación.

Como incrementar la flexibilidad de la arquitectura mediante las técnicas de reflectividad

La arquitectura en estudio (Banksphere) apoyada con técnicas de reflectividad o “Arquitectura Reflectiva” proporcionaría fundamentalmente una nueva forma de pensar acerca del proceso de cómputo. En esta arquitectura reflectiva, un sistema de software se ve como una pieza (objeto) acompañada de una pieza reflectiva. La tarea de este objeto es solucionar problemas y obtener información sobre el dominio externo, mientras que la tarea del nivel reflectivo es solucionar problemas y obtener información sobre el proceso del objeto.

En una arquitectura reflectiva uno puede asociar temporalmente la computación reflectiva con un programa de tal manera que durante la interpretación de este programa se registre una traza. Supongamos que una sesión con un sistema basado en reglas tiene que ser rastreado de tal modo que la secuencia de las reglas que se aplican requiere ser impresa. Esto se resuelve de manera sencilla con el lenguaje basado en arquitectura reflectiva indicando por ejemplo la siguiente regla reflectiva:

SI una regla tiene la prioridad más alta en una situación.
ENTONCES imprime la regla y los datos que corresponden con sus condiciones

Sin embargo, en un lenguaje basado en reglas que no está basado en una arquitectura reflectiva, el mismo resultado solamente puede ser logrado modificando el código del intérprete (de tal modo que imprima la información sobre las reglas que se aplica), o reescribiendo todas las reglas de la manera que impriman su información siempre que se apliquen.

Esta arquitectura reflectiva proporcionaría un medio para implementar computación reflectiva de una manera más modular. Como es sabido, la modularidad genera sistemas más manejables, más legibles y fáciles de entender y modificar. Pero éstas no son las únicas ventajas de la descomposición. Lo que es aún más importante es que hace posible la introducción de abstracciones que facilitan la programación de la computación reflectiva, como las abstracciones del control de estructuras, de la misma forma que el DO y WHILE facilitan la programación de flujo del control.

Así, la arquitectura se describe como un conjunto de objetos que pueden ser utilizados como cualquier otro objeto del sistema. La flexibilidad y adaptabilidad del entorno se maximizan.

Características sustanciales de la nueva arquitectura desde el punto de vista de incremento de la flexibilidad:

Con esta nueva arquitectura Reflectiva, conseguimos:

- **Objeto como única abstracción en la arquitectua**. La única entidad existente en la arquitectura es el objeto, sea cual sea su granularidad.
- **Modelo de objetos intencionadamente estándar**, con las características más comunes de lenguajes de programación más extendidos: encapsulación, herencia y polimorfismo.

- **Modo de trabajo exclusivamente OO.** Los objetos pueden crear clases que hereden de otras, crear objetos de una clase y enviar mensajes a otros objetos.

- **Objetos homogéneos**. Todos los objetos tienen la misma consideración, no hay objetos especiales que reciban un trato distinto. Los objetos propios del SO no son diferentes del resto de objetos de usuario.

- **Transparencia**. Los objetos deben ser ajenos a la existencia de mecanismos del SO que permiten conseguir características como persistencia, envío de mensajes, distribución, etc.

- **Objetos autocontenidos**. Los objetos deben tener un alto grado de autonomía, y su comportamiento no debe depender de otros objetos. Toda la información acerca del objeto está encapsulada en el mismo. Esto facilita la consecución con transparencia de objetivos como la persistencia, la distribución, etc. y decanta el modelo de objetos a un modelo de objetos activo, en el que el objeto encapsula su computación.

- **Semántica completa**. Los objetos tienen toda la semántica que da el modelo de objetos, no son considerados como una mera zona de memoria contigua.

- **Identidad de los objetos**. Cada objeto tiene un identificador único, que se usará como referencia para acceder al mismo.

CONCLUSIONES

- La reflectividad computacional es un mecanismo utilizado para otorgar flexibilidad en la semántica computacional de la arquitectura en su conjunto.

- La reflectividad supone más una técnica de lenguajes de programación que de ingeniería del software o desarrollo de aplicaciones. Los lenguajes de programación reflectivos permiten modificar o acceder, en un determinado grado, a un conjunto de sus propias características o de sus aplicaciones, utilizando para ello distintos mecanismos de implementación.

- La arquitectura Banksphere, como entorno de desarrollo de aplicaciones empresariales para el mundo web, presenta características importantes en cuanto seguridad, multicanalidad y escalabilidad. Sin embargo, presenta problemas de interoperabilidad y desadaptación de impedancias que pueden solventarse mediante técnicas de reflectividad o como parte de un sistema integral orientado a objetos.

BIBLIOGRAFÍA Y REFERENCIAS

- Reflective Computation in Objetct-Oriented Concurrent Systems and Its Applications
  Takuo Watanabe and Akinori Yonezawa. Departament of Science, Tokio Institute of Technology

- Concepts and Experiments in Computational Reflection.
  Vrije Universiteit Brussel

- Developing a Reflective Model of Collaborative Systems
  Paul Dourish. Rank Xerox Research Centre

- Implementing the Essence of Reflection: a Reflective Run-Time Environment
  Massimo Ancona. DISI – University of Genova, Genova, Italy

- Computacional Reflection in Class based Object Oriented Languages.
  Jacques Ferber. Université Paris 6, T 45-46

- Banksphere 3.0.2 v.01. Product Overview
  Banesto – Grupo SCH

- Reflectividad Computacional
  Francisco Ortín Soler. http://www.di.uniovi.es/reflection/lab

- Sistemas Integrados Orientados a Objetos.
  Daría Álvarez Gutiérrez. Universidad de Oviedo