

Tesis de grado en Ingeniería en Informática

Utilización de programación orientada a aspectos en aplicaciones enterprise

Tesista: Nicolás M. Paez

Directora: Lic. Rosita Wachenchauzer

Diciembre 17, 2007

Agenda

- Introducción
- Programación orientada a aspectos
- Aplicaciones enterprise
- Hacia AE con AOP
 - Mitos
 - Estado del arte
 - Incumbencias técnicas y del negocio
- Caso de estudio
- Conclusiones



Agenda

- Introducción
 - Programación orientada a aspectos
 - Aplicaciones enterprise
- Hacia AE con AOP
 - Mitos
 - Estado del arte
 - Incumbencias técnicas y del negocio
 - Caso de estudio
- Conclusiones

F

I

U

B

A



Problemática abordada

- La mayor parte del desarrollo de software local está abocado a aplicaciones enterprise.
 - Soporte a proceso de negocio
 - Modificaciones repentinas
 - Atributos de calidad de difícil modularización

¿y si probamos con AOP?

F
I
U
B
A



Objetivo de la tesis

- Analizar el uso de AOP en las AE
- Proponer una solución AOP para AE en la plataforma .NET
- Desarrollar una aplicación de referencia

F

I

U

B

A



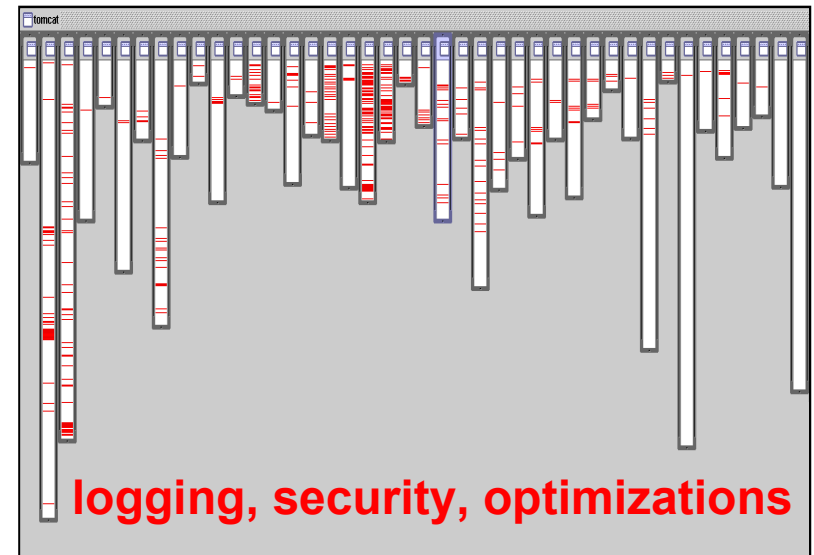
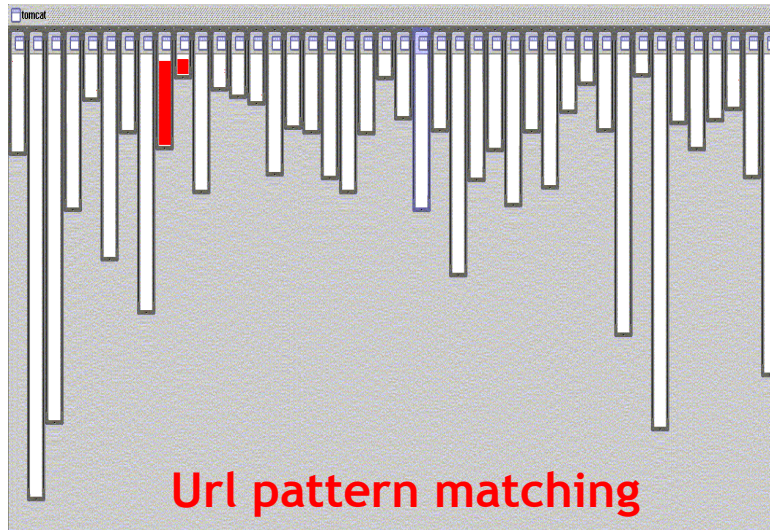
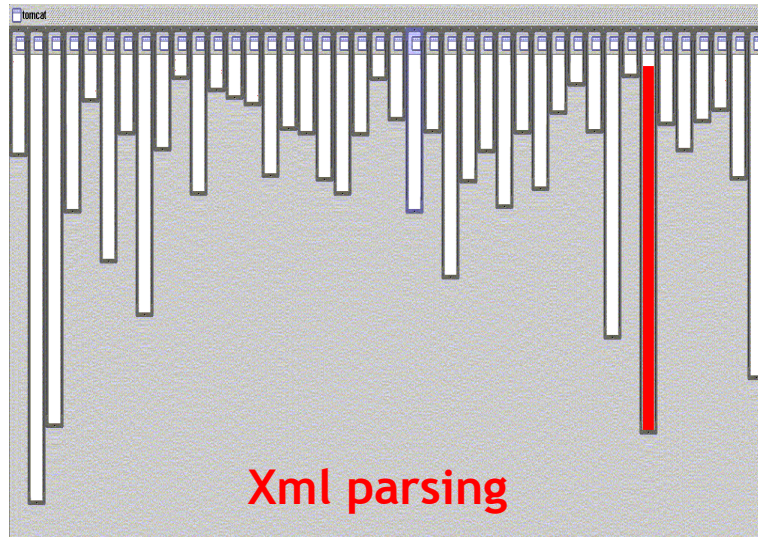
El surgimiento de AOP

- Kiczales @ XPARC 97
 - Código disperso
 - Código mezclado
 - CCC = Incumbencias transversales
 - AOP (enfoque asimétrico)
- Tarr & Harrison @ IBM 99
 - Tiranía de la descomposición dominante
 - MSOC (enfoque simétrico)
- AOSD.net



Incumbencias transversales: caso Tomcat

F
I
U
B
A



La visión AOP

En todo programa P,
ante la condición C
ejecutar la acción A.

C: pointcut, A: advice

C + A : aspecto



Elementos de AOP

- Joinpoint
- Pointcut
- Advice
- Aspecto
- Weaving

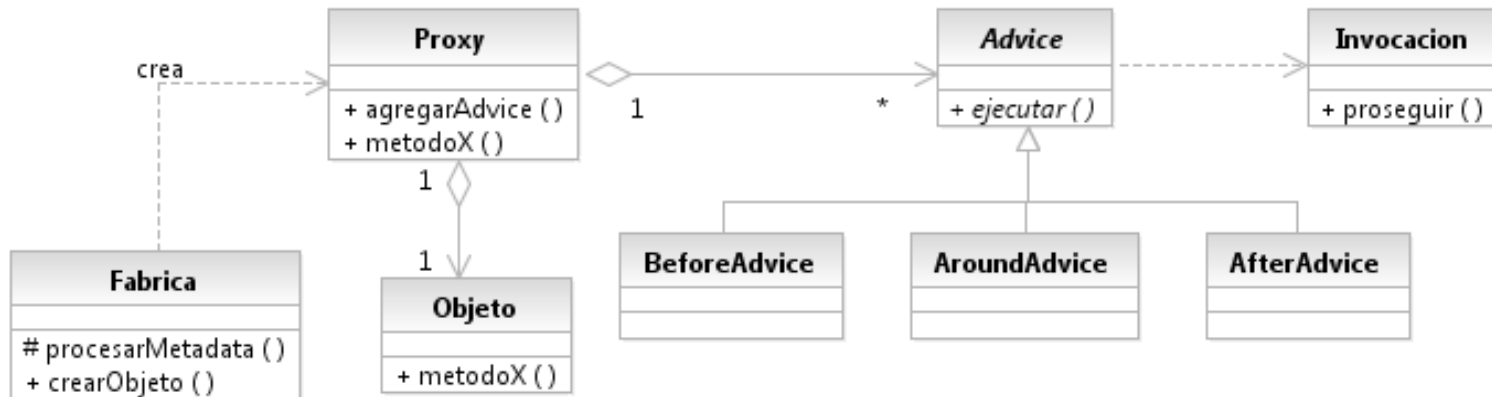
```
public class Foo {  
    public static void main(String[] args) {  
        Foo foo = new Foo();  
        foo.printFoo();  
    }  
    public void printFoo()  
    {  
        System.out.println("Foo!");  
    }  
}
```

```
public aspect LogAspect {  
    pointcut printing() : call(* *.print*(..));  
    before(): printing(){  
        System.out.println("Before printing");  
    }  
}
```



Herramientas AOP

- De entretejido estático
 - Compiladores, preprocesadores, posprocesadores
 - Aspectj, aspectC++, aspect.NET
- De entretejido dinámico
 - Basadas en proxies dinámicos
 - Spring, Naspect, JBossAOP



Estado del arte en AOP

- No hay más de 10 libros en Amazon
- Escaso uso en la industria y generalmente para monitoreo
- Líneas de investigación
 - Herramientas
 - Early aspects
 - Aplicaciones



Aplicaciones enterprise

- Soporte a procesos de negocio
- Muchos usuarios
- Grandes volúmenes de información
- Integración con otras aplicaciones
- Requisitos complejos y variantes



Aplicaciones enterprise

- Atributos de calidad
 - Disponibilidad, desempeño, escalabilidad, mantenibilidad, reusabilidad, usabilidad, seguridad,...
- Atributos técnicos
 - Transaccionalidad, caché, persistencia, distribución, monitoreo, concurrencia,

F

I

U

B

A



Estado del arte en AE

- Tecnologías
 - Java y .NET
 - Base de datos relacionales
 - Servidores de aplicaciones
 - Frameworks y librerías
- Patrones de diseño
 - Arquitectura en Capas
 - Orientación a servicios

F

I

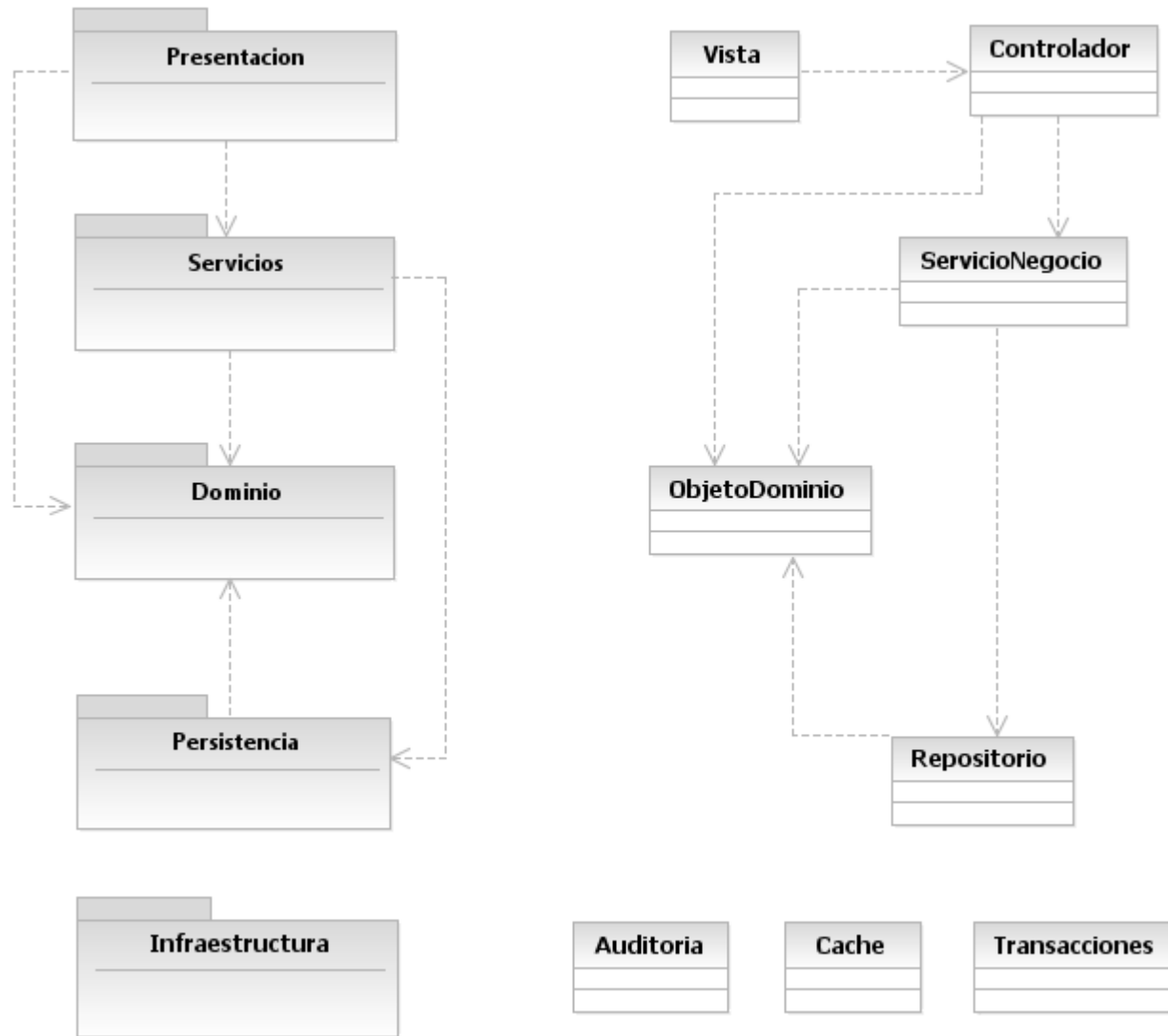
U

B

A



Arquitectura Enterprise



F
I
U
B
A



Agenda

- Introducción
 - Programación orientada a aspectos
 - Aplicaciones enterprise
- Hacia AE con AOP
 - Mitos
 - Estado del arte
 - Incumbencias técnicas y del negocio
 - Caso de estudio
- Conclusiones

F

I

U

B

A



AE & AOP

- Separación de incumbencias
 - Incumbencias técnicas
 - Incumbencias del negocio
- R. Johnson, J2EE without EJB
- R. Pawlak, AOP for J2EE
- AOP Alliance
 - Centrado en Java
 - Descontinuado
 - Poca adopción



Mitos sobre AOP

- AOP Considered harmful
- AOP viola el encapsulamiento
- Depurar con AOP es difícil
- Los patrones de diseño reemplazan AOP
- AOP = AspectJ



La piedra en el zapato de AOP

- Modelo de pointcuts basado en el código base
 - Dificulta la evolución
- Posibles soluciones
 - Herramientas de desarrollo
 - SetPoint: modelo de pointcuts semánticos

F

I

U

B

A

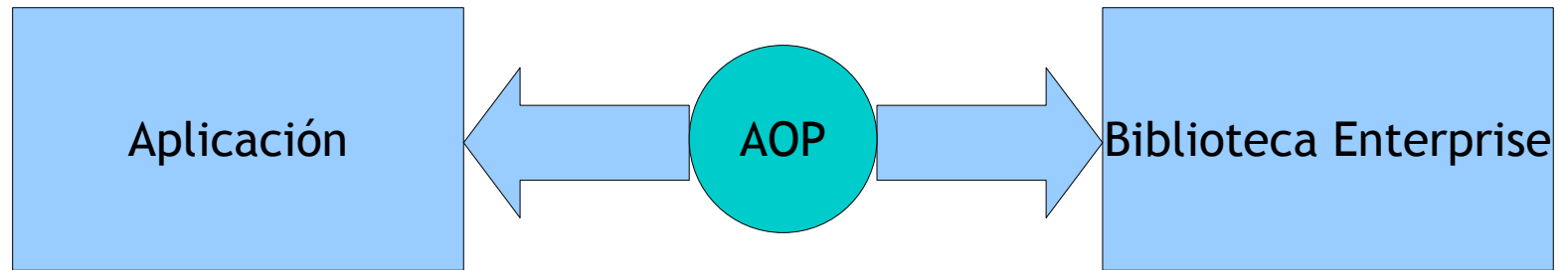


Incumbencias técnicas

- No pertenecen a la dimensión dominante
- Resulta dificultosa su modularización
- Pueden estar presentes en más de una capa
 - Auditoría
 - Caché
 - Manejo de excepciones



Incumbencias técnicas con AOP



- Entretejido dinámico
 - Permite agregar / quitar aspectos “en caliente”
 - Alto desacople entre la aplicación y la biblioteca



Incumbencias técnicas analizadas

- Monitoreo: logging, tracing y profiling
- Manejo de excepciones y Diseño por contratos
- Seguridad: **autenticación**, autorización, auditoría, confidencialidad
- **Distribución**
- Transaccionalidad
- Caché
- Concurrencia y sincronización
- **Persistencia**

F
I
U
B
A



Transaccionalidad

```
public Transferencia TransferirFondos(Cuenta cuentaOrigen, Cuenta
    cuentaDestino, decimal monto)
{
    Banco banco = new Banco();
    Transferencia transferencia = null;

    using (TransactionScope scope = new TransactionScope())
    {
        transferencia = banco.TransferirFondos(cuentaOrigen,
            cuentaDestino, monto);
        _repositorioCuentas.Actualizar(cuentaOrigen);
        _repositorioCuentas.Actualizar(cuentaDestino);
        scope.Complete();
    }
    return transferencia;
}
```

```
[Transaccional]
public Transferencia TransferirFondos(Cuenta cuentaOrigen, Cuenta
    cuentaDestino, decimal monto)
{
    Banco banco = new Banco();
    Transferencia transferencia = null;
    transferencia = banco.TransferirFondos(cuentaOrigen,
        cuentaDestino, monto);
    _repositorioCuentas.Actualizar(cuentaOrigen);
    _repositorioCuentas.Actualizar(cuentaDestino);
    return transferencia;
}
```



Transaccionalidad con AOP

```
public class TransaccionAdvice : IAroundAdvice
{
    private TransactionScopeOption _scopeOption;

    public object Invoke(IMethodInvocation invocation)
    {
        object resultado = null;
        using(TransactionScope scope = new TransactionScope(_scopeOption))
        {
            resultado = invocation.Proceed();
            scope.Complete();
        }
        return resultado;
    }
}
```

```
<aspect name="Transaction">
    <advice type="AspectosComunes.TransaccionAdvice">
        <property name="ScopeOption" value="REQUIRED" />
    </advice>
    <pointcut methodAnnotation="Transaccional" />
</aspect>
```



Caché

```
public class RepositorioCuentas : IRepositoryCuentas
{
    public IList<Cuenta> ObtenerCuentasPorCliente(Cliente cliente)
    {
        string consulta = "from Cuenta c where c.Cliente.Id=" + cliente.Id;
        IList<Cuenta> cuentas = Cache.Get(consulta) as IList<Cuenta>;
        if (cuentas == null)
        {
            cuentas = Mapper.Find<Cuenta>(consulta);
            Cache.Agregar(cliente, valor);
        }
        return cuentas;
    }
}
```

```
public class RepositorioCuentas : IRepositoryCuentas
{
    public IList<Cuenta> ObtenerCuentasPorCliente(Cliente cliente)
    {
        string consulta = "from Cuenta c where c.Cliente.Id=" + cliente.Id;
        cuentas = Mapper.Find<Cuenta>(consulta);
        return cuentas;
    }
}
```



Caché con AOP

```
public class CacheAdvice : IAroundAdvise
{
    public object Invoke(IMethodInvocation invocation)
    {
        ClaveCache clave = new ClaveCache(invocation.StaticPart,
            invocation.Arguments);

        object valor = Cache.Get(clave);

        if (valor == null)
        {
            valor = invocation.Proceed();
            Cache.Agregar(clave, valor);
        }
        return valor;
    }
}
```

```
<aspect name="Cache">
  <advice type="AspectosComunes.CacheAdvice" />
  <pointcut match="call" class="Repositorio*" method="Obtener*" />
</aspect>
```



Incumbencias del negocio

- Lógica del dominio
 - Estable, común a un conjunto de aplicaciones
- Lógica del negocio (reglas)
 - Volátil (menos estable), específica de cada organización
 - Validaciones (precondiciones)
 - Notificaciones (poscondiciones)

F

I

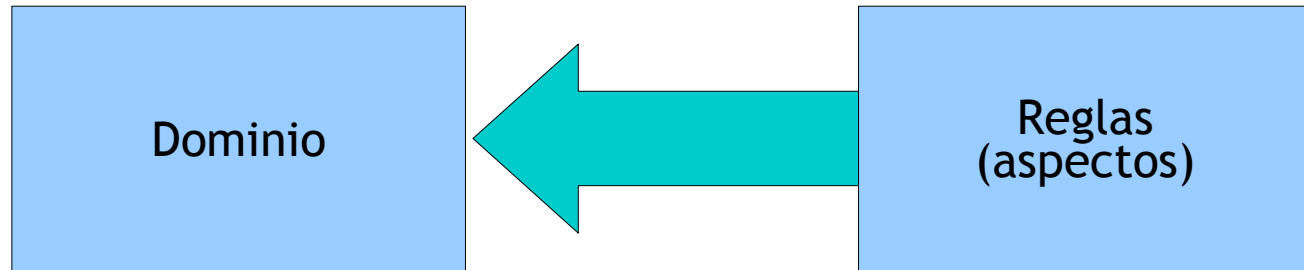
U

B

A



Incumbencias del negocio con AOP

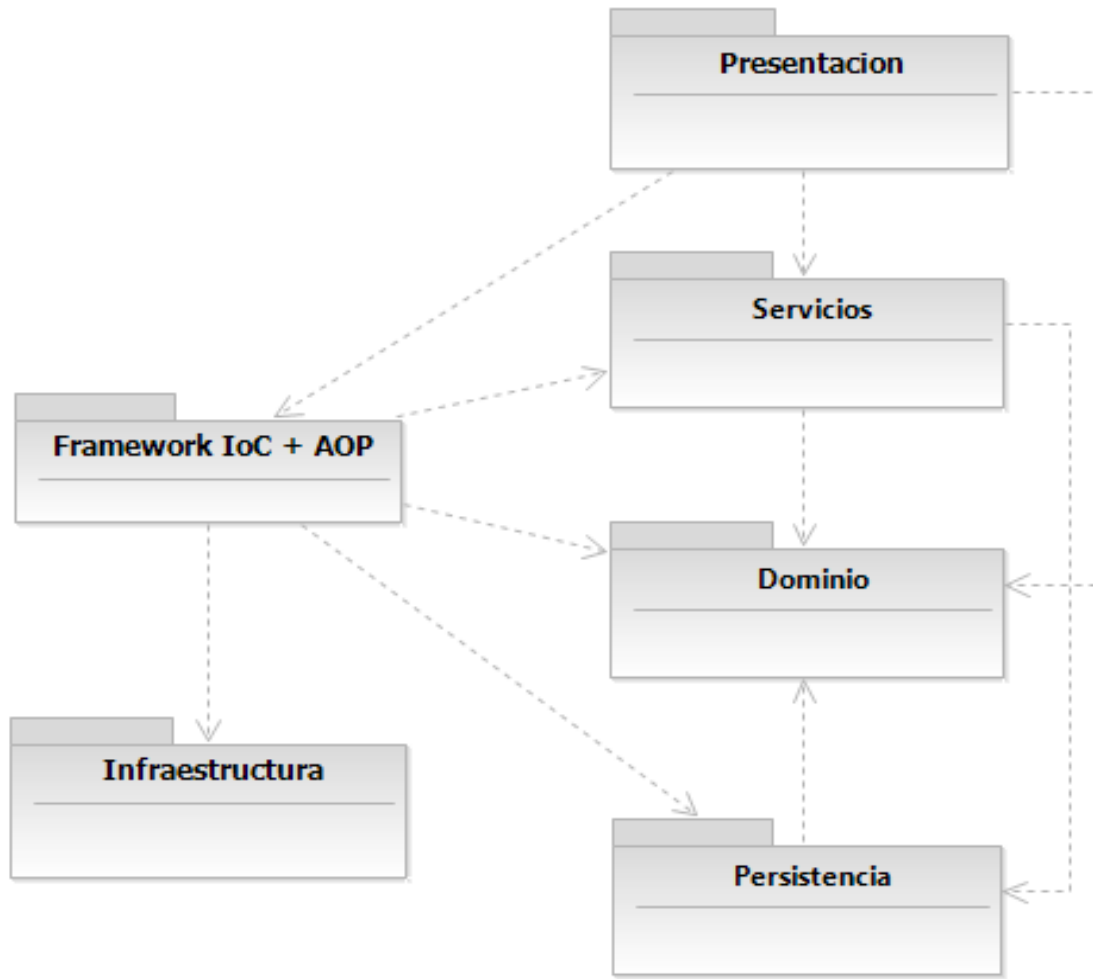


- Entretejido estático

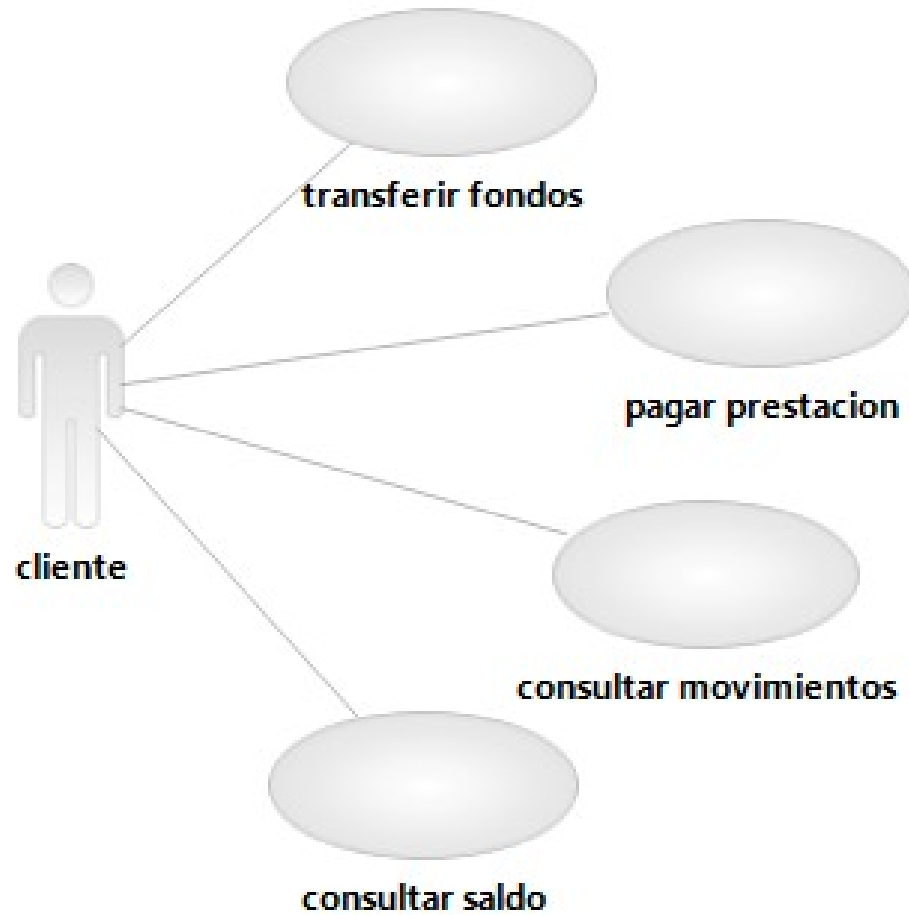
- Asegura la ejecución de las reglas (no permite que se “apaguen”)
- Pueden existir dependencias regla-dominio



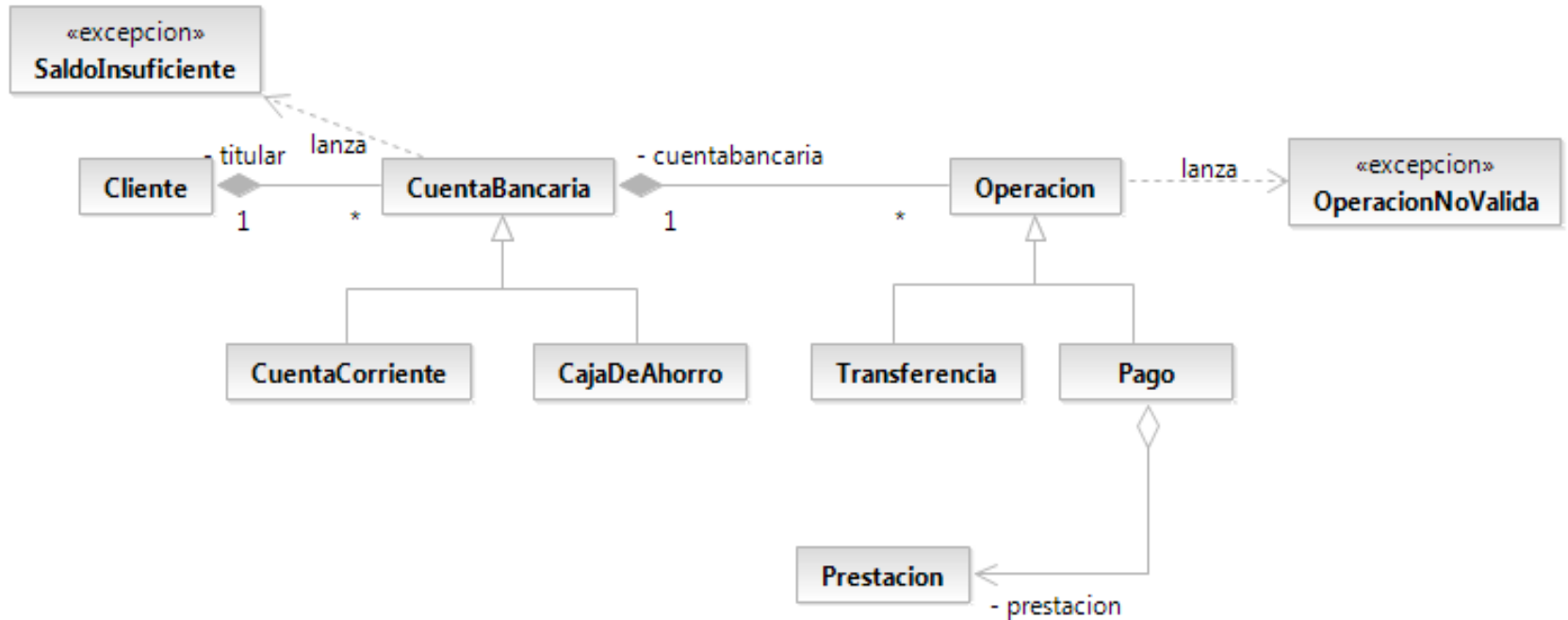
Aquitectura AOP-Compatible



Caso de estudio Banco X



Dominio Bancario



F
I
U
B
A

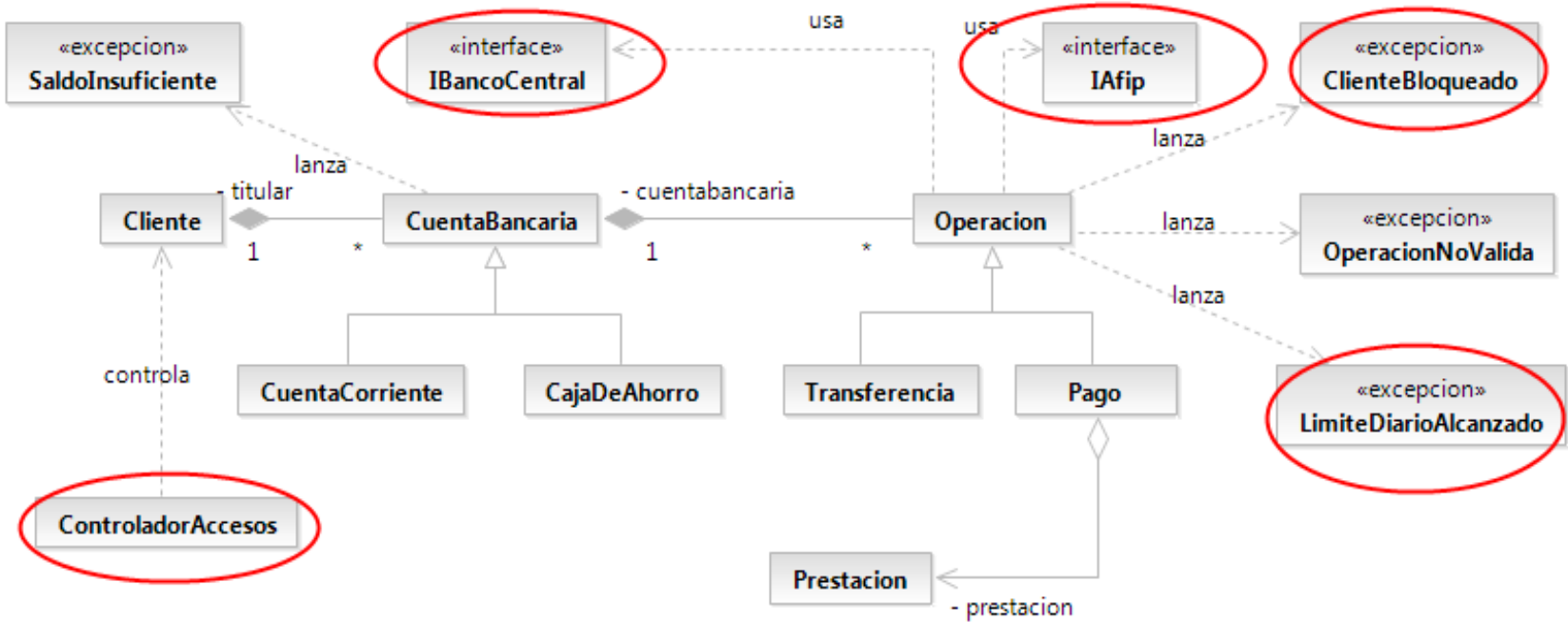


Reglas del Banco X

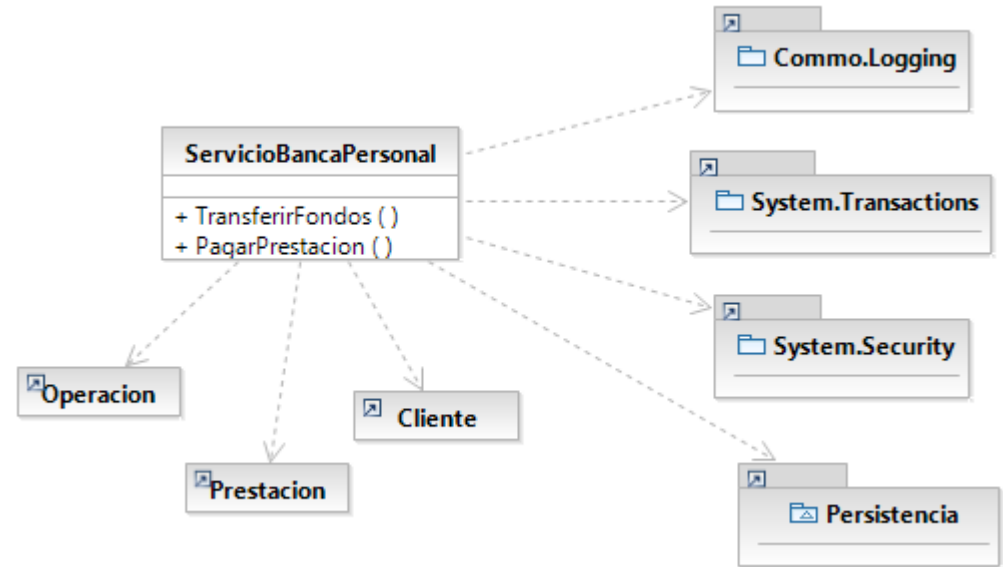
- Toda operación que supere cierto monto deberá ser notificada al Banco Central.
- No debe ser posible realizar operaciones sobre cuentas cuyos titulares tengan deudas con las Afip.
- Ante el tercer intento consecutivo de ingreso fallido al sistema por parte de un cliente, dicho cliente deberá quedar bloqueado de forma permanente sin poder acceder al sistema.
- Existe un límite diario del monto total por el cual un cliente puede realizar operaciones.
- La aplicación debe dejar constancia de todas las acciones realizadas por el cliente.



Solución sin AOP



Solución sin AOP



```
public override void Ejecutar()
{
    if (!Origen.Moneda.Equals(Destino.Moneda))
        throw new OperacionNoValidaException("Monedas distintas");

    if (_afip.TieneDeuda(Destino.Titular) || _afip.TieneDeuda(Origen.Titular))
        throw new ClienteSuspendidoPorDeudasException();

    VerificarLimiteDiario(_origen.Titular, Monto);

    Origen.Debitar(Monto);
    Destino.Accreditar(Monto);

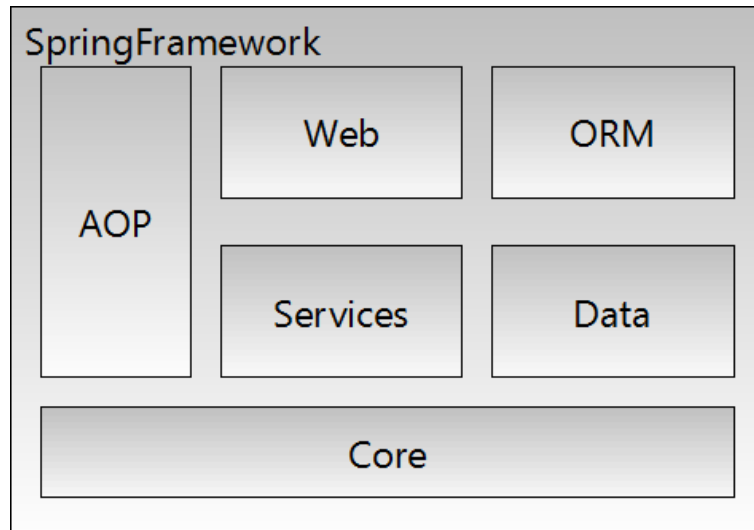
    _bancoCentral.NotificarOperación(this);

    FechaHora = DateTime.Now;
    Numero = Secuencia;
}
```

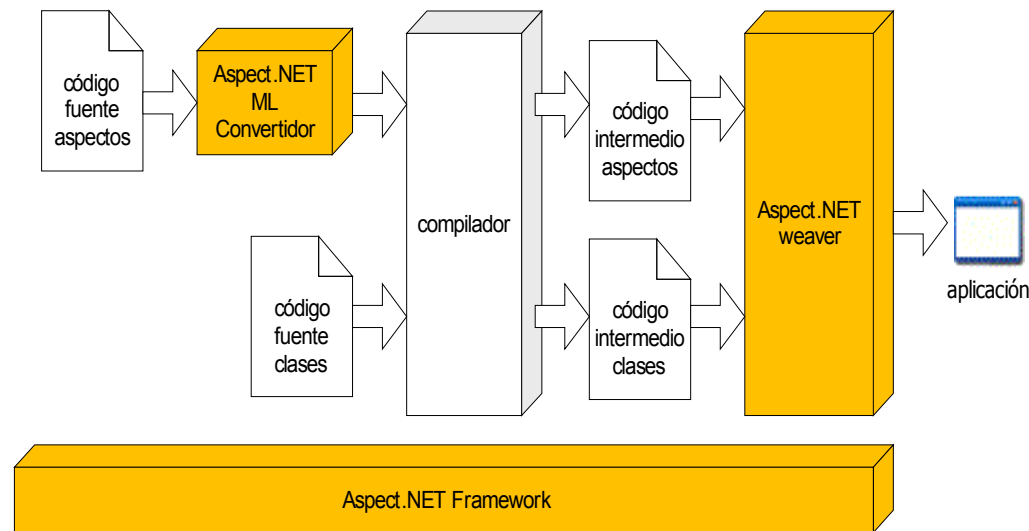


Herramientas AOP utilizadas

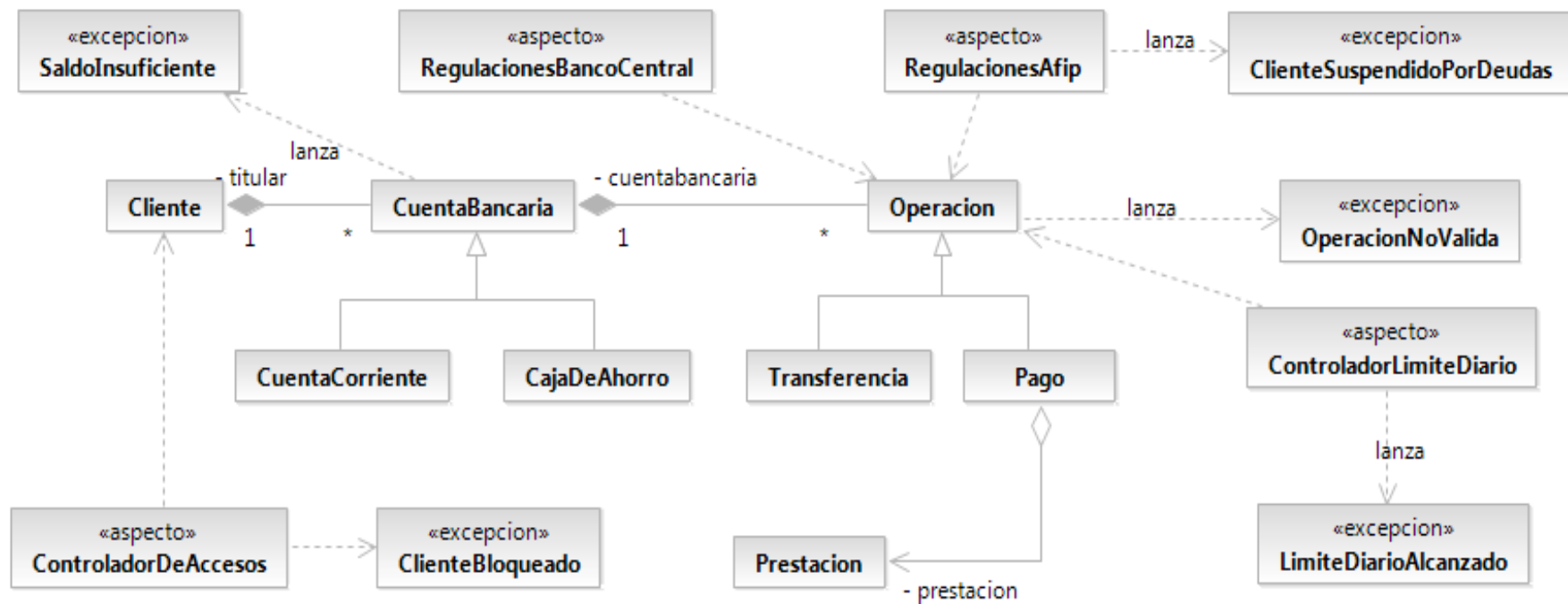
- Spring AOP
(entretejido dinámico)



- Aspect.NET
(entretejido estático)



Solución AOP



Agenda

- Introducción
 - Programación orientada a aspectos
 - Aplicaciones enterprise
- Hacia AE con AOP
 - Mitos
 - Estado del arte
 - Incumbencias técnicas y del negocio
 - Caso de estudio
- Conclusiones

F

I

U

B

A



Conclusiones I

- Incubencias técnicas
 - tienen naturaleza transversal y el uso de AOP resulta muy conveniente
 - Facilita el foco en el negocio
 - Permite en alto grado de reuso de aspecto
- Incumbencias de negocio
 - AOP permite separar las reglas específicas del negocio de los conceptos generales del dominio, facilitando la evolución de las reglas



Conclusiones II

- Caso de estudio
 - Mejor separación de incumbencias
 - Reducción de código repetido, disperso y entremezclado
- Herramientas utilizadas
 - Aspect.NET aún le falta maduración respecto a AspectJ
 - Es necesaria mayor integración con el entorno de desarrollo (asistentes, depuración, etc)

F
I
U
B
A



Conclusiones III

- Conceptualmente falta la posibilidad de indicar donde NO deben aplicarse aspectos.
- Cuestiones abiertas
 - Uso de AOP en la capa de presentación
 - Estudio de las posibilidades de separación de incumbencias ofrecidas por la versión 3.5 de la plataforma.NET
 - Actividades de ingeniería orientada a aspectos

F
I
U
B
A



Fin

Nicolás Paez

npaez@fi.uba.ar

www.fi.uba.ar/~npaez

Sitios de referencia

- AspectJ: www.eclipse.org/aspectj
- Spring: www.springframework.net
- AOSD: www.aosd.org
- Early Aspects: www.earlyaspects.org



Back up slides

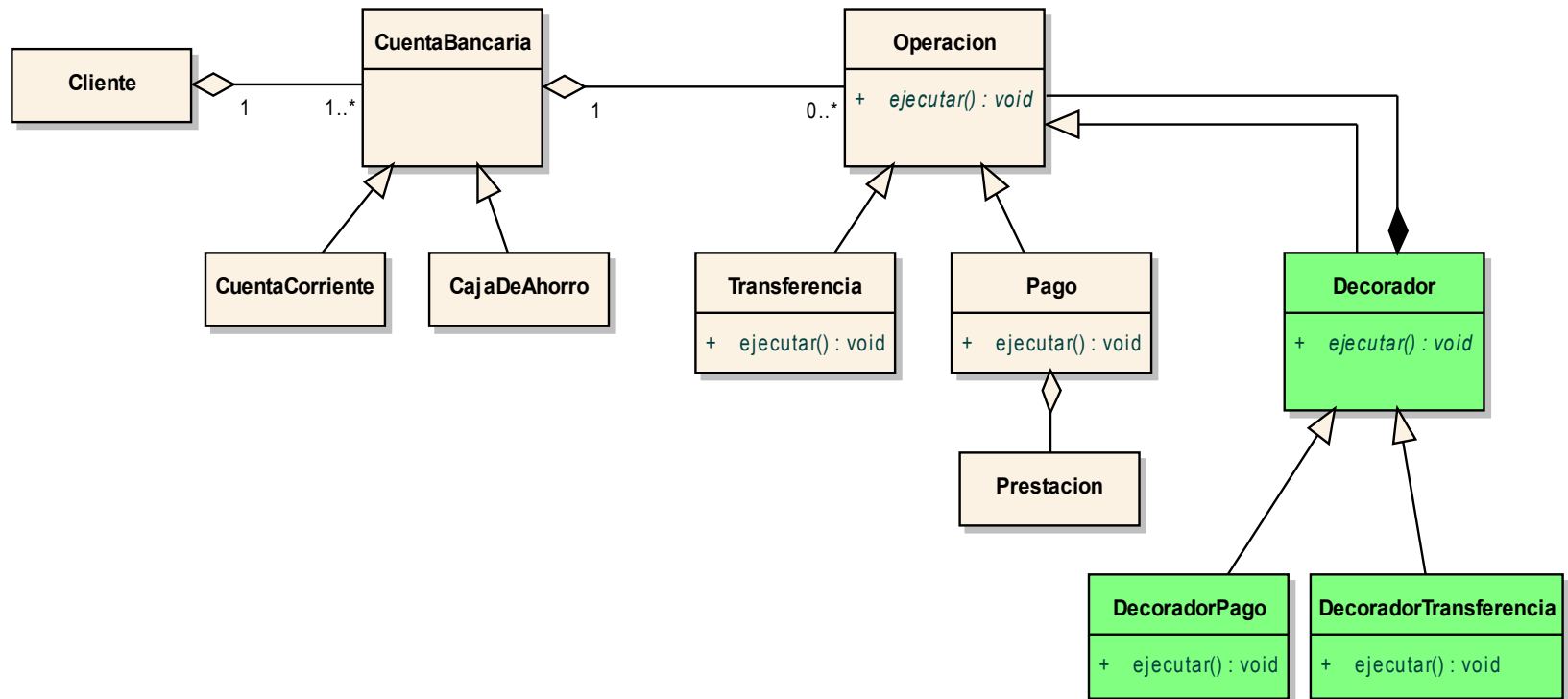
Definiciones

- Requisito: circunstancia o condición necesaria para algo.
- Requirimiento: acción y efecto de requerir. Aviso, manifestación o pregunta que se hace, generalmente bajo fe notarial, a alguien exigiendo o interesando de él que exprese y declare su actitud o su respuesta
- Requirement:
 - necesidad. Choose a car that suits your requirements = Elija un coche adecuado a sus necesidades. What are your requirements? = ¿qué necesita usted?. To meet somebody's requirements = satisfacer las necesidades de alguien.
 - requisito. You must satisfy these requirements = debe llenar or satisfacer estos requisitos

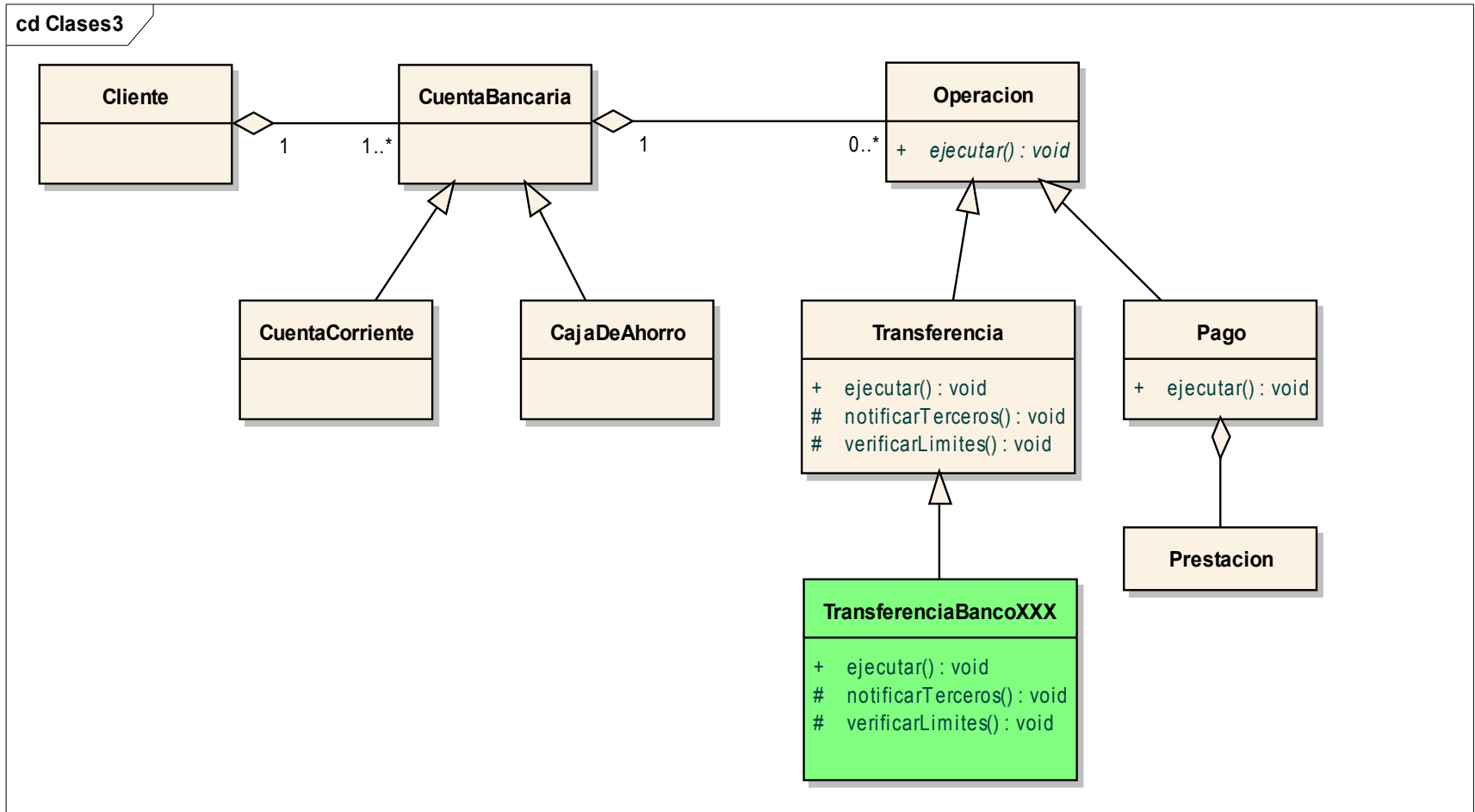


Alternativas con Patrones: Decorator

cd Clases

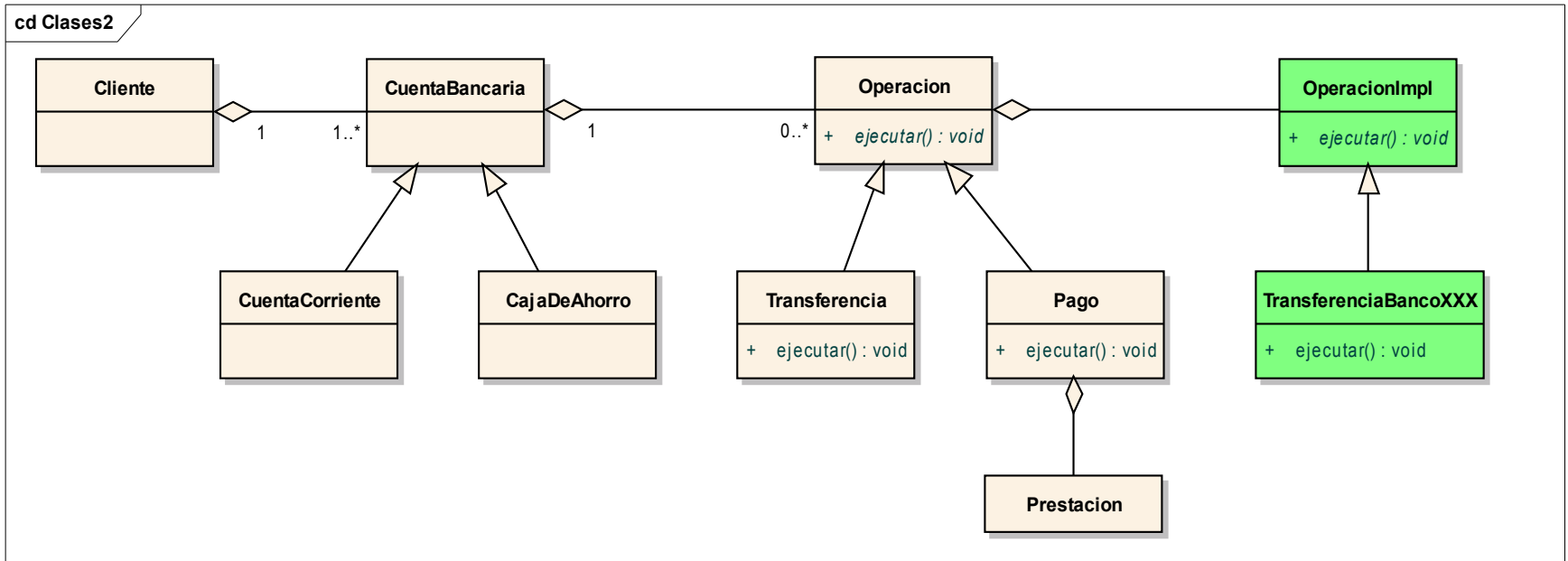


Aternativas con patrones: template method

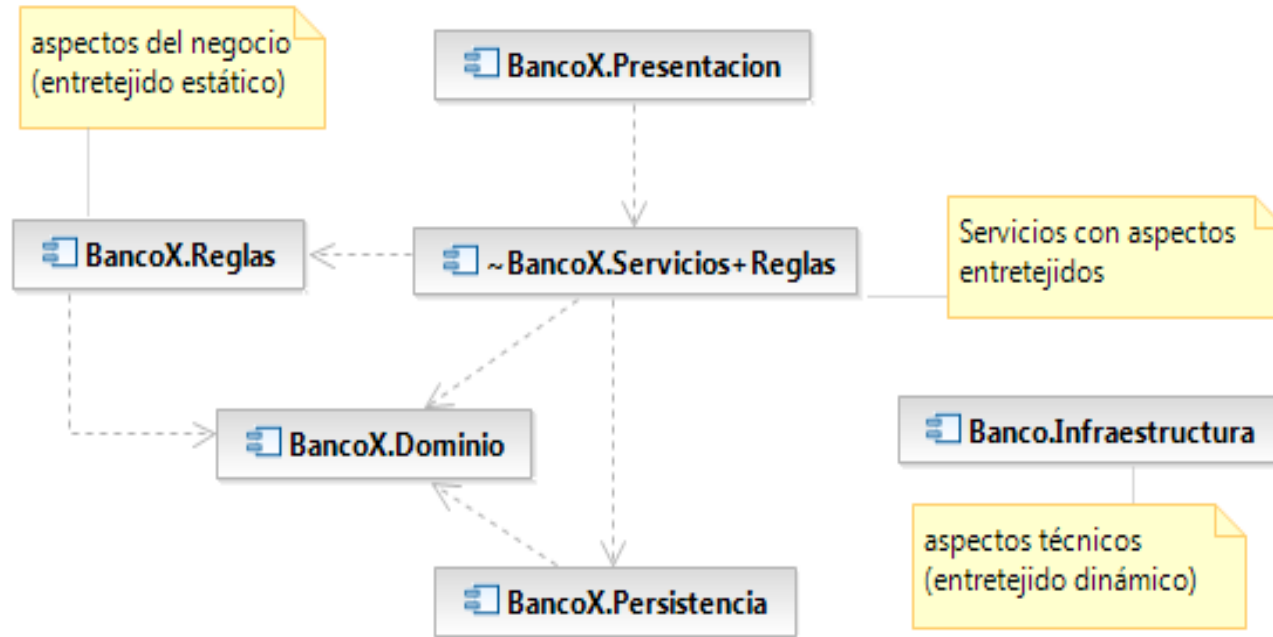


Alternativas con patrones: bridge

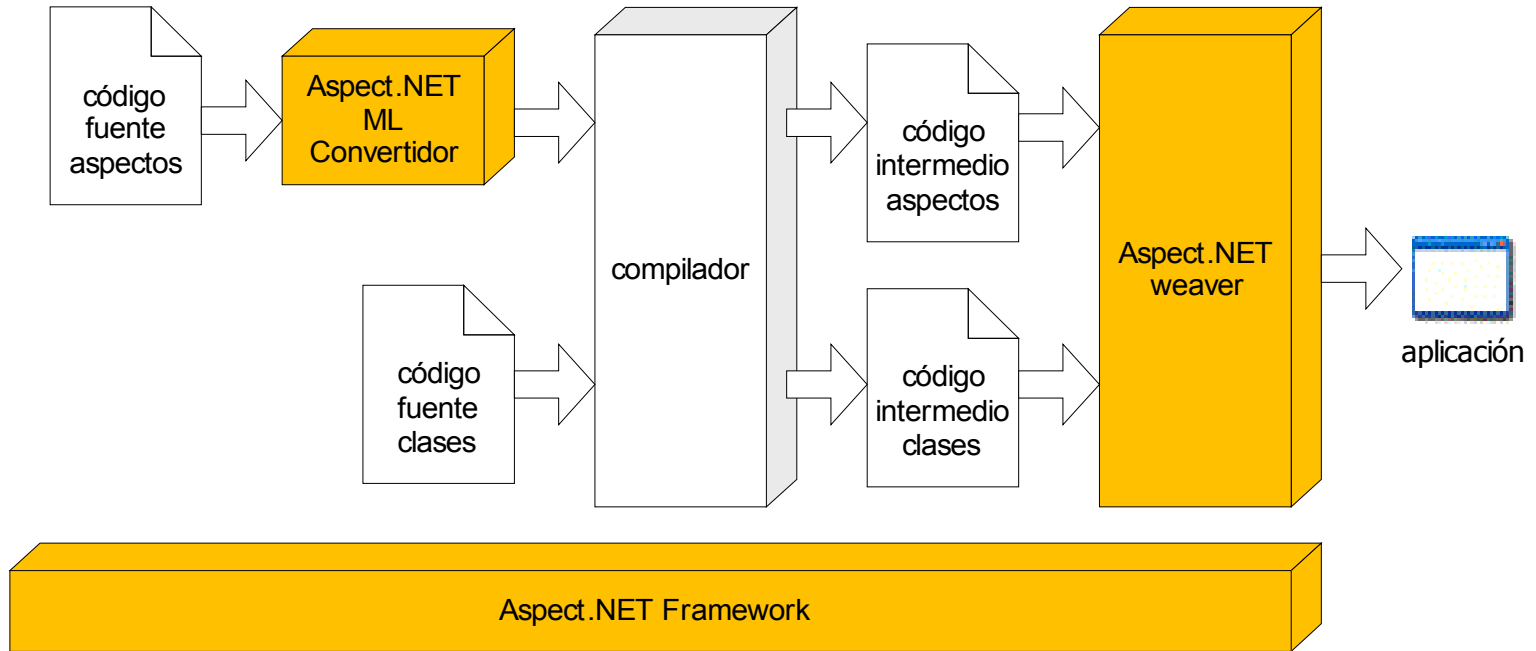
F
I
U
B
A



Componentes de la solución AOP



Aspect.NET



F
I
U
B
A



Spring Framework

