Tesis Final

Título

Manipulación de objetos persistentes en clientes de Aplicaciones Enterprise

Alumnos

Bottero, Sebastian Eduardo

García, Nicolás Javier

Director

Gustavo Rossi

Tesis Final

Manipulación de objetos persistentes en clientes de Aplicaciones Enterprise

Temario

- **Introducción**
- Contexto
- Análisis del Problema
- Solución
- Contribuciones

Introducción

Sistemas de información Enterprise

- Conjunto de sistemas de computación que permiten a una empresa u organización:
  - **Integrar y coordinar** sus procesos de negocios
  - **Compartir información** por todos los niveles de la jerarquía de administración
  - Eliminar los problemas originados por la **fragmentación de la información** que se genera al utilizar **múltiples sistemas** de información

Introducción

Aplicaciones Enterprise

- Son parte fundamental de los **sistemas de Información Enterprise**
- Sus principales objetivos a satisfacer son:
  - Mejorar sus procesos en eficiencia y productividad
  - Unificar la manera de relacionar la información, sus recursos humanos, tecnológicos y de infraestructura

Introducción

- En este tipo de aplicaciones encontramos distintos **objetos** que modelan conceptos del dominio donde:
  - Su volumen de información es elevado y por ende son naturalmente **profundos y de gran tamaño**
  - Deben ser **persistentes** en el tiempo

Introducción

- Son accedidos de manera **concurrente**
- Son consultados y/o modificados por **cientos de usuario**
- Interactúan con gran cantidad de **interfaces de usuario**
- Deben ser manejados dentro de un **tiempo de respuesta** razonable

Objetivos

Analizar los Sistemas Enterprise e identificar los **problemas recurrentes** y **requerimientos no funcionales** que surgen de implementar clientes para este tipo de aplicaciones sobre una arquitectura Cliente Servidor

Objetivos

Definir una manera de **especificar información** sobre el comportamiento de las instancias de los objetos de dominio en la componente cliente, para ser luego **interpretada**

Objetivos

Diseñar un framework para **interpretar la información** y resolver cada uno de los problemas analizados, que se producen al **manipular** los objetos del dominio en el cliente de Aplicaciones Enterprise, para su visualización, utilización, modificación y posterior persistencia de los cambios.

Temario

- Introducción
- **Contexto**
  - **Requerimientos funcionales y no funcionales**
  - Arquitectura de Software
  - Transferencia del modelo de Entidades
- Análisis del Problema
- Solución
- Contribuciones

Requerimientos Funcionales

- Derivan mayormente del dominio al cual está orientada la empresa u organización
- Las Aplicaciones Enterprise por lo general tienen como base las siguientes características:
  - La información debe estar integrada
  - Manejo de objetos del negocio de gran tamaño
  - Log de operaciones
  - Multimoneda

Requerimientos No Funcionales

- Como consecuencia de las **características** inherentes a las Aplicaciones Enterprise de gran tamaño y de los **requerimientos funcionales**, aparecen los requerimientos no funcionales

Requerimientos No Funcionales

- **Performance**
- Capacidad y Escalabilidad
- **Usabilidad**
- Adaptabilidad al cambio de reglas de negocio
- Portabilidad a través de plataformas
- Seguridad
  - Control de acceso, contraseña única y encriptación
- Alta Disponibilidad
- **Mantenibilidad**

Requerimientos No Funcionales

- Hot deploy
- Autonomía del usuario
- Descentralización de operaciones
- Modos de Operación
  - Normal y mantenimiento
- **Uso de Red**
- Facilidad de Instalación
  - Múltiples ambientes
- Integración con sistemas legacy

Temario

- Introducción
- **Contexto**
  - Requerimientos funcionales y no funcionales
  - **Arquitectura de Software**
  - Transferencia del modelo de Entidades
- Análisis del Problema
- Solución
- Contribuciones

Arquitectura de software

- Si bien existen muchas definiciones, se puede observar que hay dos elementos que se repiten
  - **Descomposición de alto nivel** que divide al sistema en sus partes
  - **Decisiones que son difíciles de cambiar**

- Contribuye a establecer los fundamentos de la **plataforma de desarrollo** a la que pertenece
- Define de manera abstracta los **componentes**, sus interfaces y la comunicación entre ellos

Clientes

- Dependiendo de capacidad de proceso que posea en comparación al servidor, se pueden clasificar en:
- Pesado (Cliente de Mail)
  - **Mayor capacidad de proceso**, interfaces nativas, almacenamiento.
- Híbridos (Nuevas tecnologías WEB)
  - Capacidad de **procesar datos enviados por el servidor**
- Livianos (Viejos navegadores WEB)
  - **Sin capacidad de procesamiento**, recoge los datos del usuario, se los envía al servidor, y mostrar una respuesta

Temario

- Introducción
- **Contexto**
  - Requerimientos funcionales y no funcionales
  - Arquitectura de Software
  - **Transferencia del modelo de Entidades**
- Análisis del Problema
- Solución
- Contribuciones

Transferencia del modelo de Entidades

- **Modelo de entidades –** Abstracciones de entidades reales o elementos internos del sistema
  - Persistentes
  - Transferidas entre el cliente y el servidor

- El objetivo es **interactuar** con una entidad transferida como se lo hace con una **local**

Transferencia del modelo de Entidades

- Interfaces
  - Grano fino -> problemas en ambiente remoto
  - Grano grueso en los límites de distribución (Servicios)
- Data Transfer Object
  - Reducción del número de llamadas remotas -> Mejora del rendimiento
  - Posible explosión de clases -> Esfuerzo adicional de codificación

Transferencia del modelo de Entidades

- Detach Objects (fuera del “contexto origen”)
  - Interactuar con objetos como en el servidor
  - Profundidad de los datos enviados - Acceso concurrente
- Dynamic Transfer Object - DyTO
  - Evitar generación de los DTO con un **modelo dinámico** (Mapa de Propiedades)
  - **Modelo declarativo** para la “población” de datos
  - **Proxy** para interactuar con interface original

Temario

- Introducción
- Contexto
- **Análisis del Problema**
  - **Concurrencia**
  - Transacciones largas
  - Objetos de gran tamaño (carga por demanda)
  - Deshacer operaciones
- Solución
- Contribuciones

Concurrencia

- Problemas al trabajar con la misma entidad al mismo tiempo
  - Se soluciona permitiendo trabajar sólo a un proceso con la misma entidad
  - Se reduce cantidad de actividad concurrente
- Soluciones
  - **Aislamiento** - **Datos inmutables**

Concurrencia (cont.)

- Imposible aislar los datos -> Políticas de Lockeo
  - **Lockeo Optimista** - “detecta” los problemas
  - **Lockeo Pesimista** - “previene” los problemas

- Elección dependiendo de la frecuencia y la severidad de los conflictos

Temario

- Introducción
- Contexto
- **Análisis del Problema**
  - Concurrencia
  - **Transacciones largas**
  - Objetos de gran tamaño (carga por demanda)
  - Deshacer operaciones
- Solución
- Contribuciones

Transacciones largas

- Transacción **de sistema**
  - Aplicación -> BD
- Transacción **de negocios**
  - Usuario -> Aplicación
- **Transacciones largas**
  - Más de un ciclo pedido/respuesta al servidor
  - Propiedades ACID

Transacciones largas (cont.)

- Una conexión de BD por transacción
  - Poca escalabilidad
- Impide el uso del mecanismo de loqueo pesimista provisto por la base de datos
- Patrón de concurrencia offline
  - Se delega el soporte ACID a la aplicación
  - Partir transacción larga en varias de sistema

Temario

- Introducción
- Contexto
- **Análisis del Problema**
  - Concurrencia
  - Transacciones largas
  - **Objetos de gran tamaño (carga por demanda)**
  - Deshacer operaciones
- Solución
- Contribuciones

Objetos de gran tamaño

- Muchos colaboradores con **grafos profundos**
- **Colecciones** con elevada cant. de elementos
- Carga de todos los objetos relacionados
  - Evita cargar objetos colaboradores de manera explicita
- **Carga por demanda**
  - Cargar objetos sólo cuando son necesarios
  - Analizar objetos que se recuperan en la misma operación

Temario

- Introducción
- Contexto
- **Análisis del Problema**
  - Concurrencia
  - Transacciones largas
  - Objetos de gran tamaño (carga por demanda)
  - **Deshacer operaciones**
- Solución
- Contribuciones

Deshacer operaciones

"El software que permite deshacer es software en el que puedes confiar"

- Beneficios del Undo
  - Alivio de las preocupaciones
  - Una forma de reducir la complejidad
  - Un método para construir la confianza
- Durante una transacción larga, y en forma previa a la confirmación de la operación, debe permitirse volver atrás con **algunos** cambios realizados.

Temario

- Introducción
- Contexto
- Análisis del Problema
- **Solución**
  - **Plataforma de desarrollo**
  - Framework propuesto
  - Arquitectura del Framework
  - Meta-Información
- Contribuciones
  - Transacciones largas
  - Manejo de concurrencia
  - Deshacer cambios
  - Referencias Lazy y carga por demanda

Plataforma de desarrollo

- Conjunto de frameworks y servicios que:
  - Proveen una **forma sencilla** y **coherente** de **integrar** las distintas **funcionalidades** a desarrollar
  - Definen los **lineamientos** y las **pautas** de desarrollo y ayudan a **estandarizarlo**
  - Dan **solución** a los distintos **problemas** que se presentan en el **proceso de desarrollo** de Aplicaciones Enterprise

Temario

- Introducción
- Contexto
- Análisis del Problema
- **Solución**
  - Plataforma de desarrollo
  - **Framework propuesto**
  - Arquitectura del Framework
  - Meta-Información
- Contribuciones
  - Transacciones largas
  - Manejo de concurrencia
  - Deshacer cambios
  - Referencias Lazy y carga por demanda

Framework Propuesto

- Resolver la **transferencia de información** entre las **capas** de la arquitectura
  - Manteniendo desacopladas la capa de presentación del resto de las capas
  - Lidiando con los problemas inherentes al manejo de objetos persistentes en el cliente
- Naturaleza estructural
  - Resuelve un problema tecnológico
  - No se origina de los requerimientos funcionales del sistema

Framework Propuesto

- Satisface una serie de requerimientos destinados a brindar una mayor flexibilidad en su uso
  - Paginado de colecciones de objetos
  - Orden y filtro en colecciones paginadas
  - Límite para colecciones
  - Chequeo de integridad

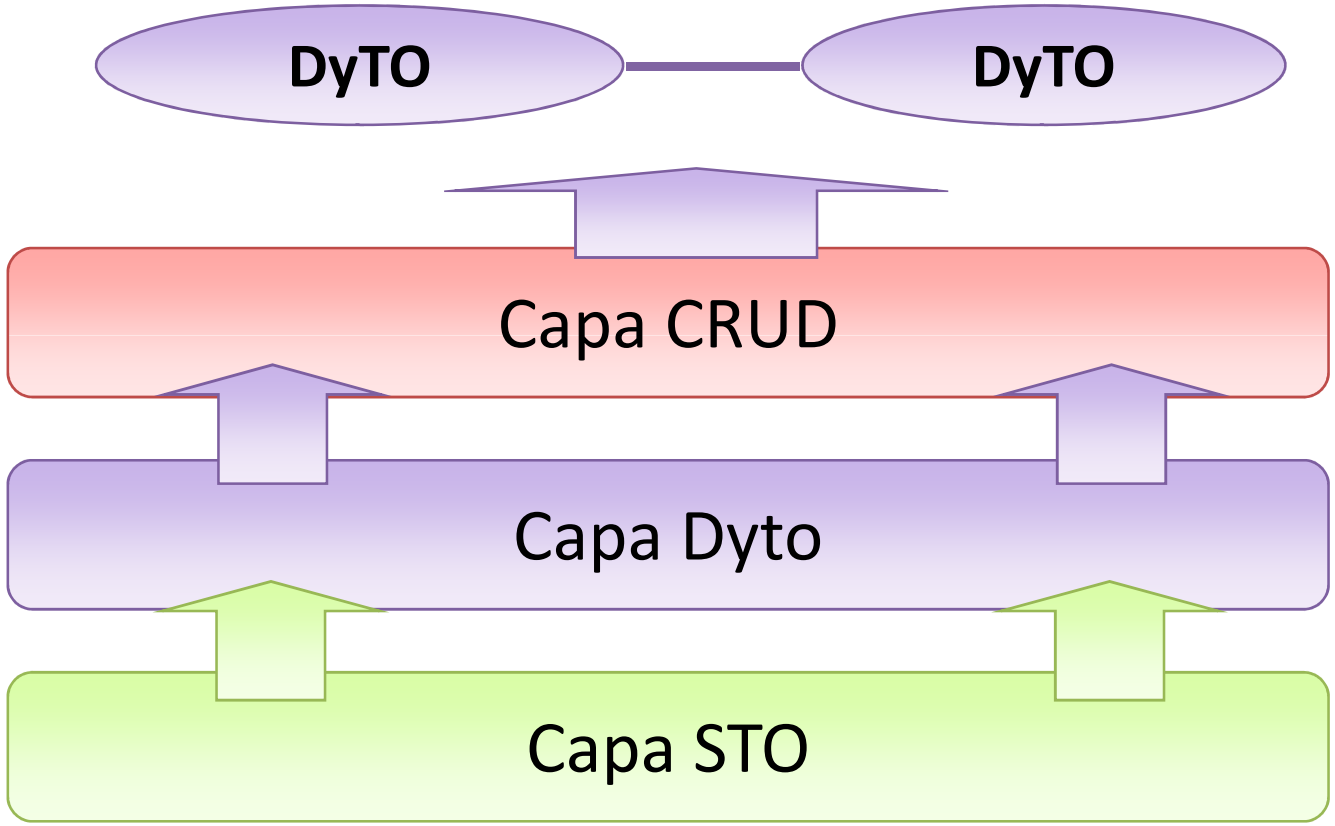
Temario

- Introducción
- Contexto
- Análisis del Problema
- **Solución**
  - Plataforma de desarrollo
  - Framework propuesto
  - **Arquitectura del Framework**
  - Meta-Información
- Contribuciones
  - Transacciones largas
  - Manejo de concurrencia
  - Deshacer cambios
  - Referencias Lazy y carga por demanda

Arquitectura del Framework

- El framework está organizado en **capas de niveles de abstracción**.
- Cada capa delega la comunicación en la capa **inmediatamente inferior**.
- Cada capa se basa en las abstracciones provistas por su inmediata inferior, permitiendo atacar **un problema cada vez**

Arquitectura del Framework (cont.)



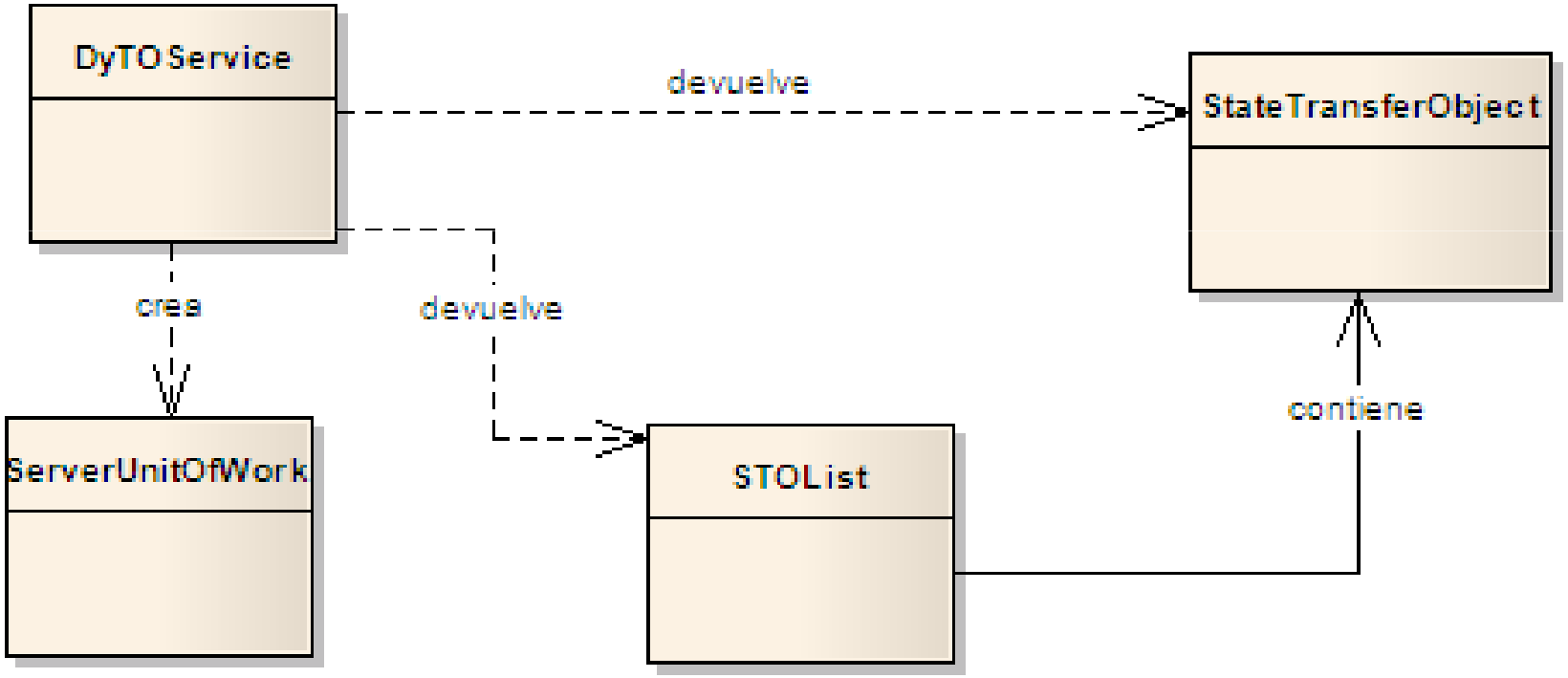
Arquitectura del Framework (cont.)

Capa de State Transfer Objects

- Se encarga de:
  - **Crear** los **objetos de transferencia** entre el servidor y el cliente a partir de **descriptores** de propiedades
  - **Modificar** un objeto de dominio mediante el procesamiento del **log de modificaciones**
- Es la que interactúa con la capa de persistencia o repositorio de objetos

Arquitectura del Framework (cont.)

Capa de State Transfer Objects



Arquitectura del Framework (cont.)

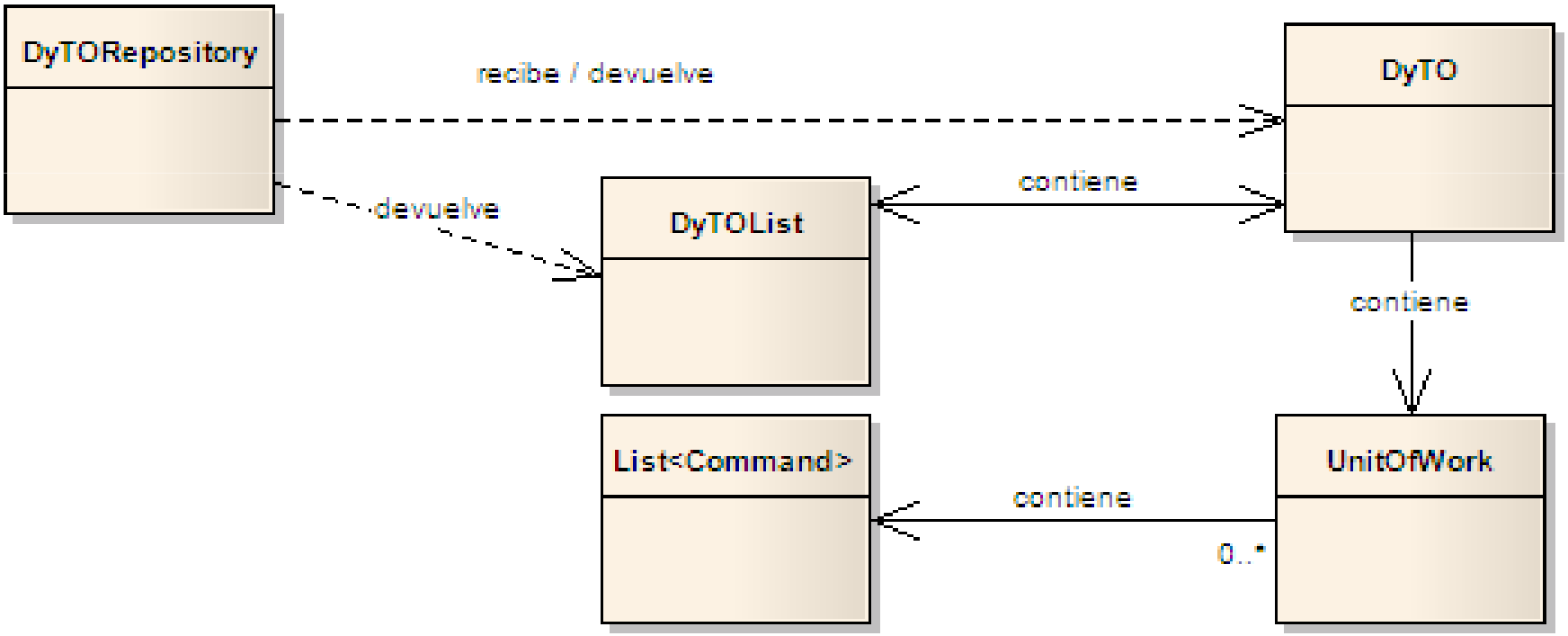
Capa de DyTOs

- Se encarga de:
  - Mantener el **log de modificaciones** realizadas
  - Resolver la carga por demanda
  - Contener atributos calculados

- Es la responsable del manejo de los objetos de transferencia de datos dinámicos (DyTOs)

Arquitectura del Framework (cont.)

Capa de DyTOs



Arquitectura del Framework (cont.)

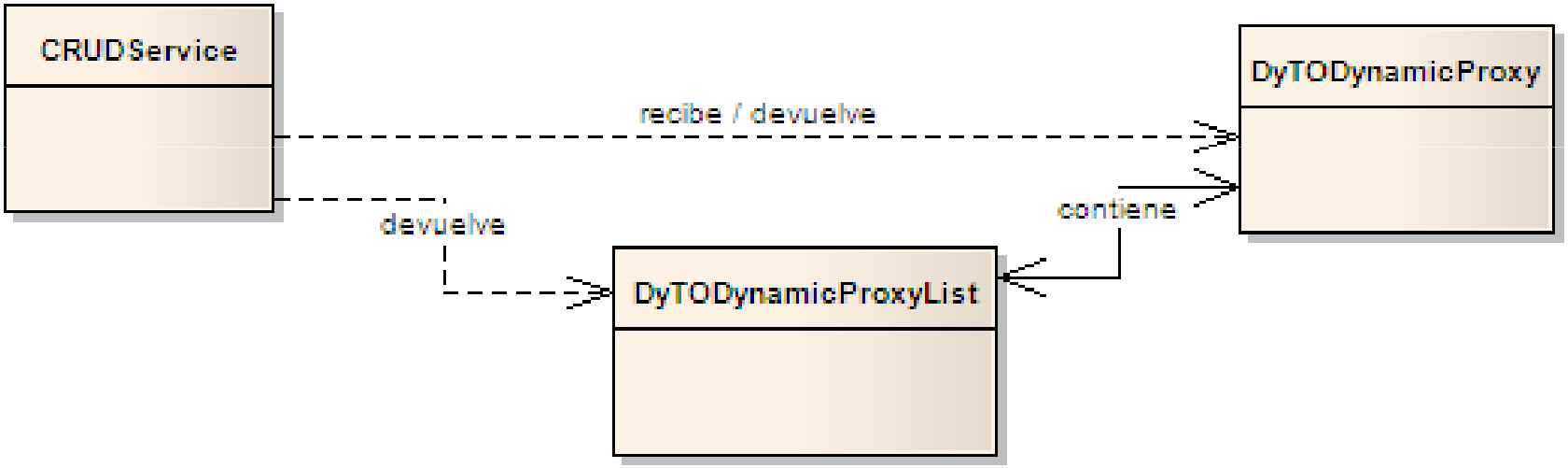
Capa de Servicios CRUD

- Se encarga de:
  - Proveer una interfaz para realizar operaciones CRUD
  - Brindar una abstracción de los objetos de transferencia

- Es la capa con la cual interactúa el usuario del framework

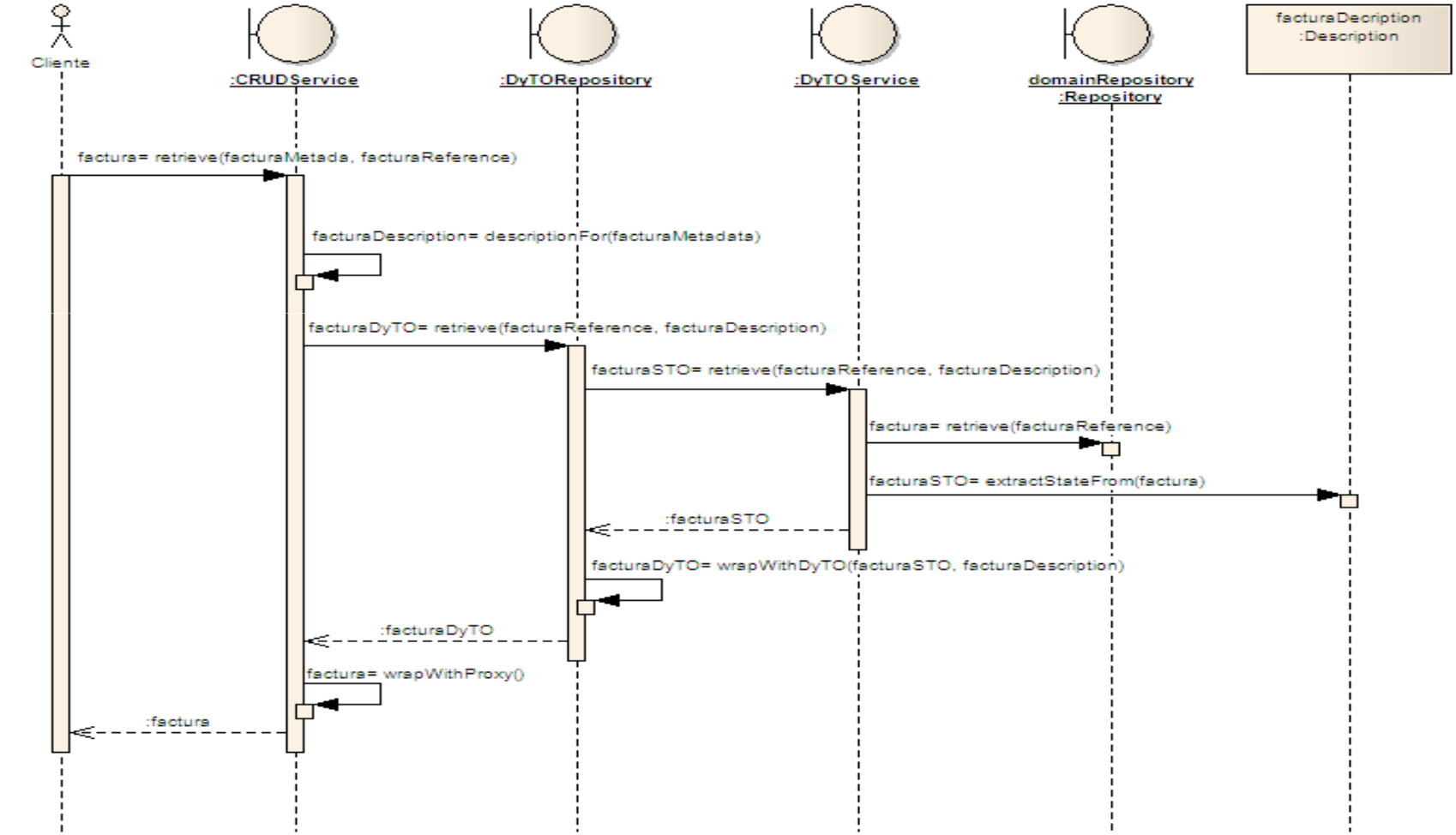
Arquitectura del Framework (cont.)

Capa de Servicios CRUD



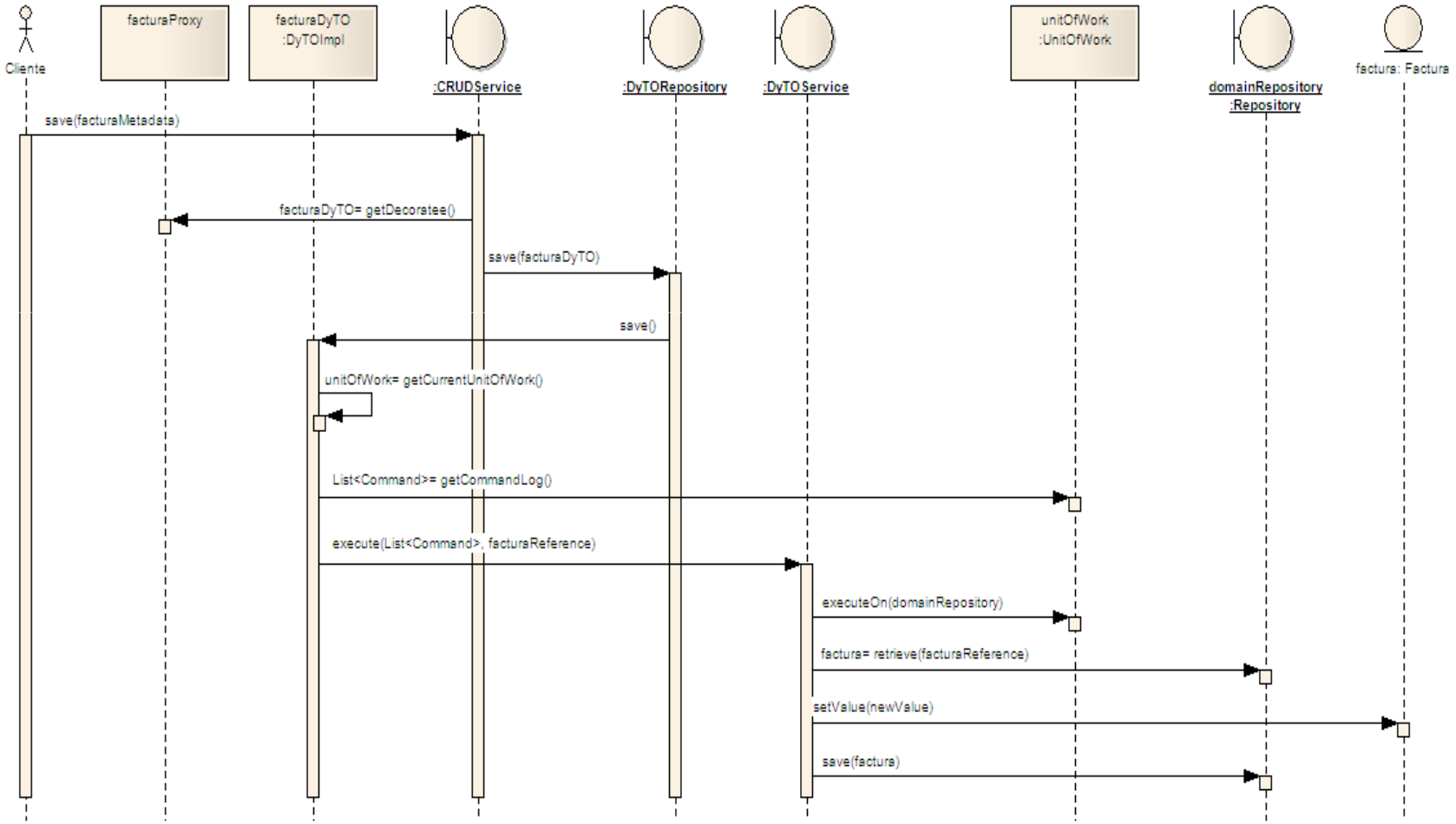
Arquitectura del Framework (cont.)

Operación RETRIEVE



Arquitectura del Framework (cont.)

Operación SAVE



Temario

- Introducción
- Contexto
- Análisis del Problema
- **Solución**
  - Plataforma de desarrollo
  - Framework propuesto
  - Arquitectura del Framework
  - **Meta-Información**
- Contribuciones
  - Transacciones largas
  - Manejo de concurrencia
  - Deshacer cambios
  - Referencias Lazy y carga por demanda

Meta-Información

- Permite indicar **cómo** y **de qué manera** hacer el mapeo de los objetos del modelo de dominio a los DyTOs que se envían al cliente
- Qué propiedades del objeto de dominio se llevan al cliente
  - Simples
  - Calculadas
  - DyTO
  - Colección

Meta-Información (cont.)

- Mecanismo para dar **información adicional** sobre cierto tipo de propiedades
  - Carga de Entidades o Colecciones por demanda (lazy)
  - Tamaño de página y orden en las colecciones
  - Etc.

- Un DyTO toma sus datos a partir **único objeto** de dominio principal, pudiendo combinar (o aplanar) información de todos los **objetos relacionados** con él

Temario

- Introducción
- Contexto
- Análisis del Problema
- **Solución**
  - Plataforma de desarrollo
  - Framework propuesto
  - Arquitectura del Framework
  - Meta-Información
  - **Transacciones largas**
  - Manejo de concurrencia
  - Deshacer cambios
  - Referencias Lazy y carga por demanda
- Contribuciones

Transacciones largas

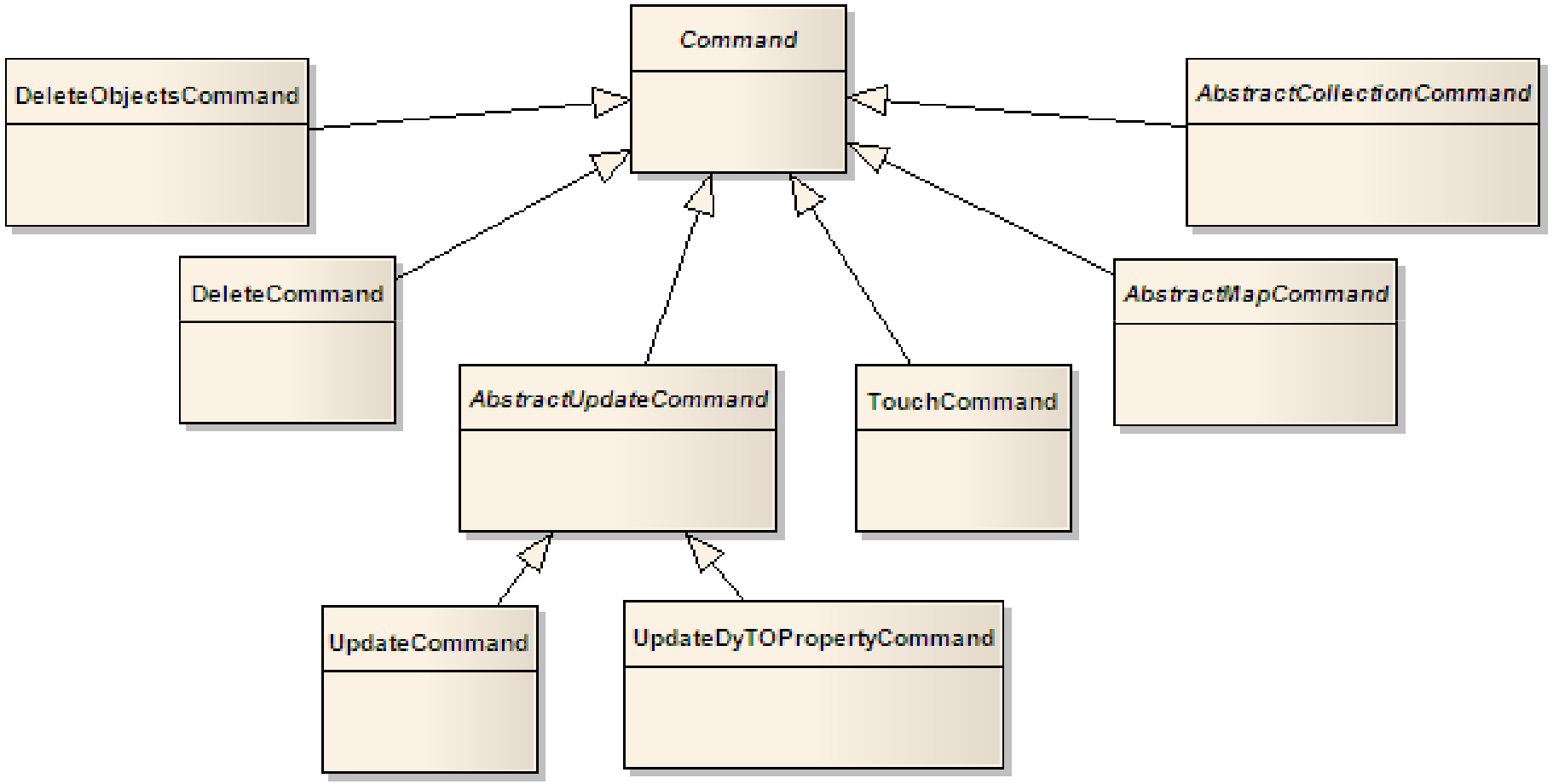
- Operaciones que tienen características de transacciones largas o de negocios, es decir, que abarcan más de un ciclo pedido-respuesta al servidor de aplicaciones

- Cada DyTO posee su propia **Unit of Work** en donde se registran todas las operaciones que se realizan sobre él, utilizando **Comandos**

Transacciones largas (cont.)

- Se **registran** todas las operaciones que se realizan
- Orden preciso en el cual cada DyTO del grafo fue modificado -> **índice único de comandos**
- Al finalizar la transacción se arma un único log -> todos los comandos ordenados según el índice
- Un tipo de comando distinto para cada tipo de operación

Transacciones largas (cont.)



Temario

- Introducción
- Contexto
- Análisis del Problema
- **Solución**
  - Plataforma de desarrollo
  - Framework propuesto
  - Arquitectura del Framework
  - Meta-Información
  - Transacciones largas
  - **Manejo de concurrencia**
  - Deshacer cambios
  - Referencias Lazy y carga por demanda
- Contribuciones

Manejo de concurrencia

- Se enriquece el mecanismo provisto por los frameworks de persistencia
  - Al modificar un objeto en el cliente, se "**marca**" el DyTO correspondiente (*TouchCommand*)
  - Por defecto, no afecta a los objetos colaboradores, salvo al agregar elementos a sus colecciones

Manejo de concurrencia (cont.)

- Lockeo Optimista
  - En el servidor se aplican los *TouchCommand* y se verifica la versión del objeto con respecto a la referencia
  - Si alguna de las versiones no concuerda, se aborta la transacción

Temario

- Introducción
- Contexto
- Análisis del Problema
- **Solución**
  - Plataforma de desarrollo
  - Framework propuesto
  - Arquitectura del Framework
  - Meta-Información
  - Transacciones largas
  - Manejo de concurrencia
  - **Deshacer cambios**
  - Referencias Lazy y carga por demanda
- Contribuciones

Deshacer cambios

- Volver atrás cambios en una transacción de negocio que involucre varias pantallas
  - El mecanismo elegido es la definición de **checkpoints** (basado en el concepto de UOW)
  - Por cada cambio en el DyTO, se generan **comandos de acciones inversas** y se guardan en el checkpoint

Deshacer cambios (cont.)

- Si se cancela, se vuelve atrás el estado de los DyTOs ejecutando los comandos inversos contenidos en el checkpoint actual
- Cuando se aceptan los cambios, el checkpoint se ignora
- Posibilidad de definir **varios checkpoints**

Temario

- Introducción
- Contexto
- Análisis del Problema
- **Solución**
  - Plataforma de desarrollo
  - Framework propuesto
  - Arquitectura del Framework
  - Meta-Información
  - Transacciones largas
  - Manejo de concurrencia
  - Deshacer cambios
  - **Referencias Lazy y carga por demanda**
- Contribuciones

Carga por demanda - Referencias Lazy

- Ventaja con respecto a implementaciones del patrón DTO
  - Propiedades con "**referencias**" a otros DyTOs
  - Se reemplazan por DyTOs construidos **bajo demanda**
  - Se invoca al servidor, se construye el DyTO y se reemplaza la referencia por el DyTO
  - Además se agrega el DyTO al Identity Map

Temario

- Introducción
- Contexto
- Análisis del Problema
- Solución
- **Contribuciones**

Contribuciones

- Se **detalló el origen** de gran parte de problemas y requerimientos no funcionales

- Se plantearon **alternativas de solución** a cada uno de estos problemas

- Se definió un framework que da **soporte**, de manera transparente, a todos estos temas utilizando soluciones "exitosas”

Contribuciones

- Además, al utilizar el framework propuesto:
  - **Estandarizamos** el desarrollo
  - Mejoramos la **productividad del desarrollo**
  - Mejoramos la **calidad de los productos** desarrollados
  - Simplificamos el futuro **mantenimiento** de las aplicaciones desarrolladas
  - **Eliminamos** la dificultad de cumplir con los requerimientos no funcionales

Trabajo Futuro

- **Lenguaje específico para meta información** (DSL)
  - Implementación del código de configuración
  - Si es externo, librería para su interpretación
- **Implementación integral del Framework**
- **Arquitectura dirigida por modelos** (MDA)
  - Modelos independientes y específicos de la plataforma (PIM y PSM)
  - Transformaciones necesarias para definir una MDA sobre la solución provista

Preguntas?