UTN FACULTAD REGIONAL CÓRDOBA
INGENIERÍA EN SISTEMAS DE INFORMACIÓN

CÁTEDRA DE DISEÑO DE SISTEMAS DE INFORMACIÓN

Asignación de Responsabilidades a los objetos

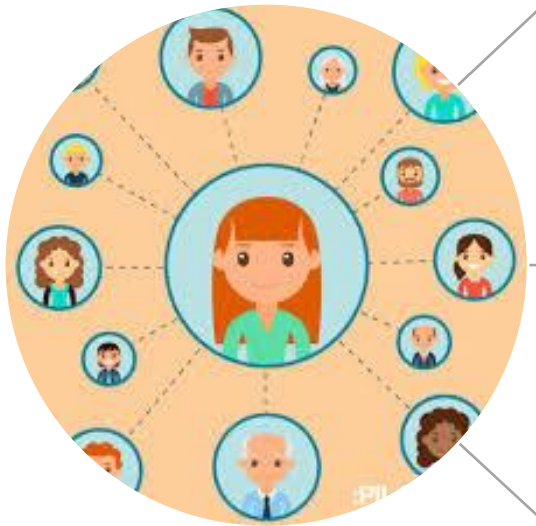
PATRONES GRASP

Para Análisis

(Craig Larman)

Patrones GRASP: Responsabilidades relacionadas con: Hacer - Conocer

- Hacer algo en uno mismo.
- Iniciar una acción en otros objetos.
- Controlar y coordinar actividades en otros objetos.

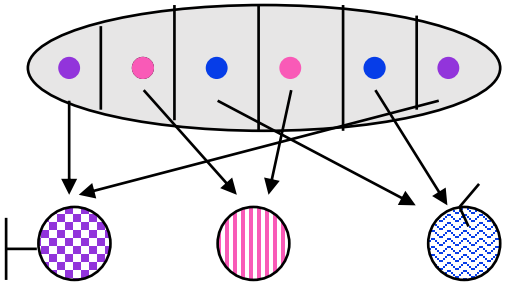


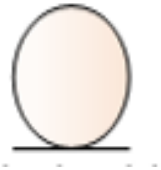
- Estar enterado de los datos privados encapsulados.
- Estar enterado de la existencia de objetos conexos.
- Estar enterado de cosas que se pueden derivar o calcular.

Patrones Grasp para Análisis

- Experto en Información
- Creador
- Bajo Acoplamiento
- Alta Cohesión
- Controlador

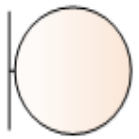
Clases del Análisis



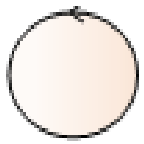


Clase de Entidad: modela información que podría ser persistente y que puede sobrevivir a una ejecución de un sistema.

Clases de Fabricación Pura

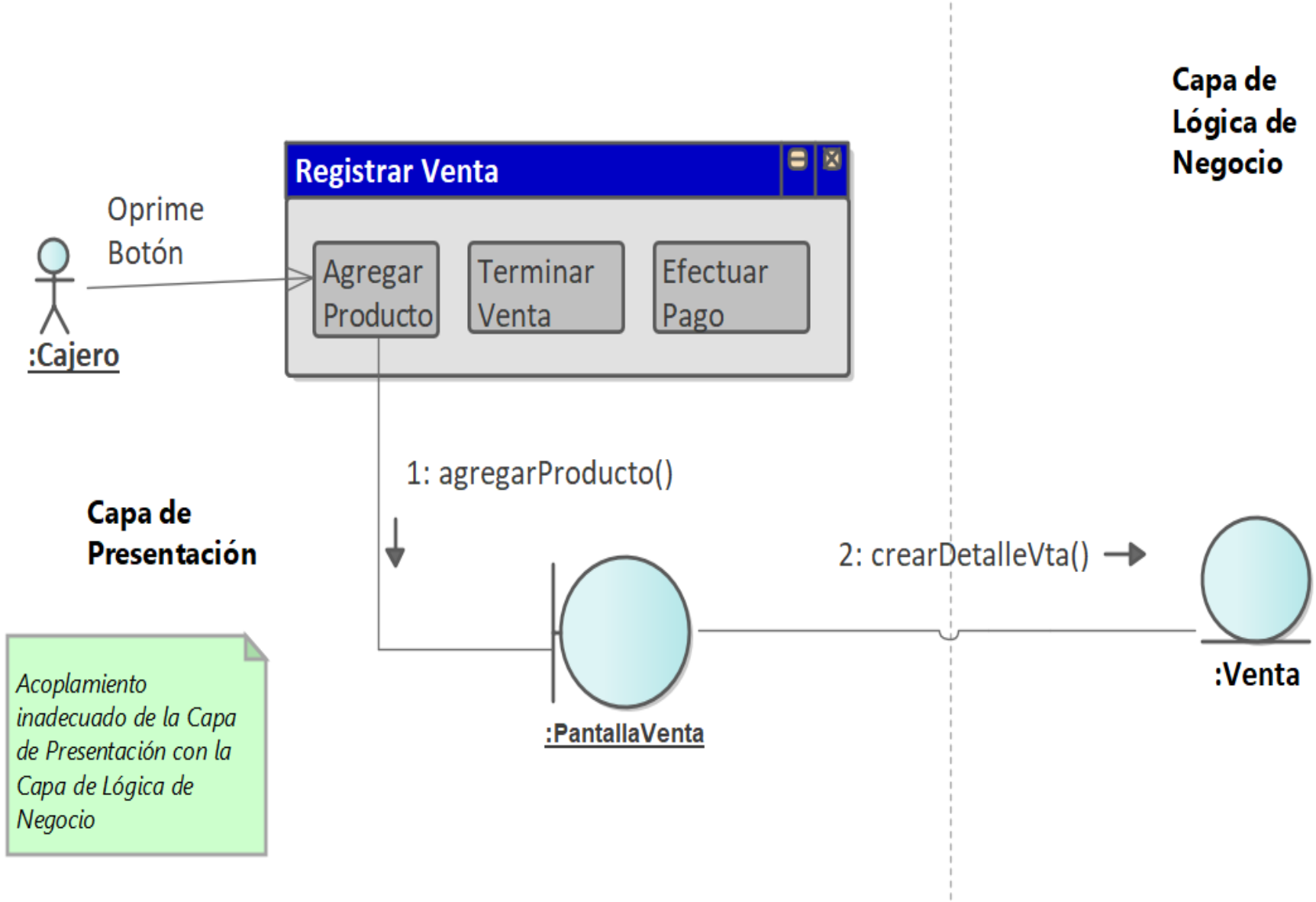


Clase límite o frontera (Boundary) modela el comportamiento e información que es dependiente de la frontera del sistema con el ambiente. Modela todo lo que concierne a cualquier vínculo sistema-actor

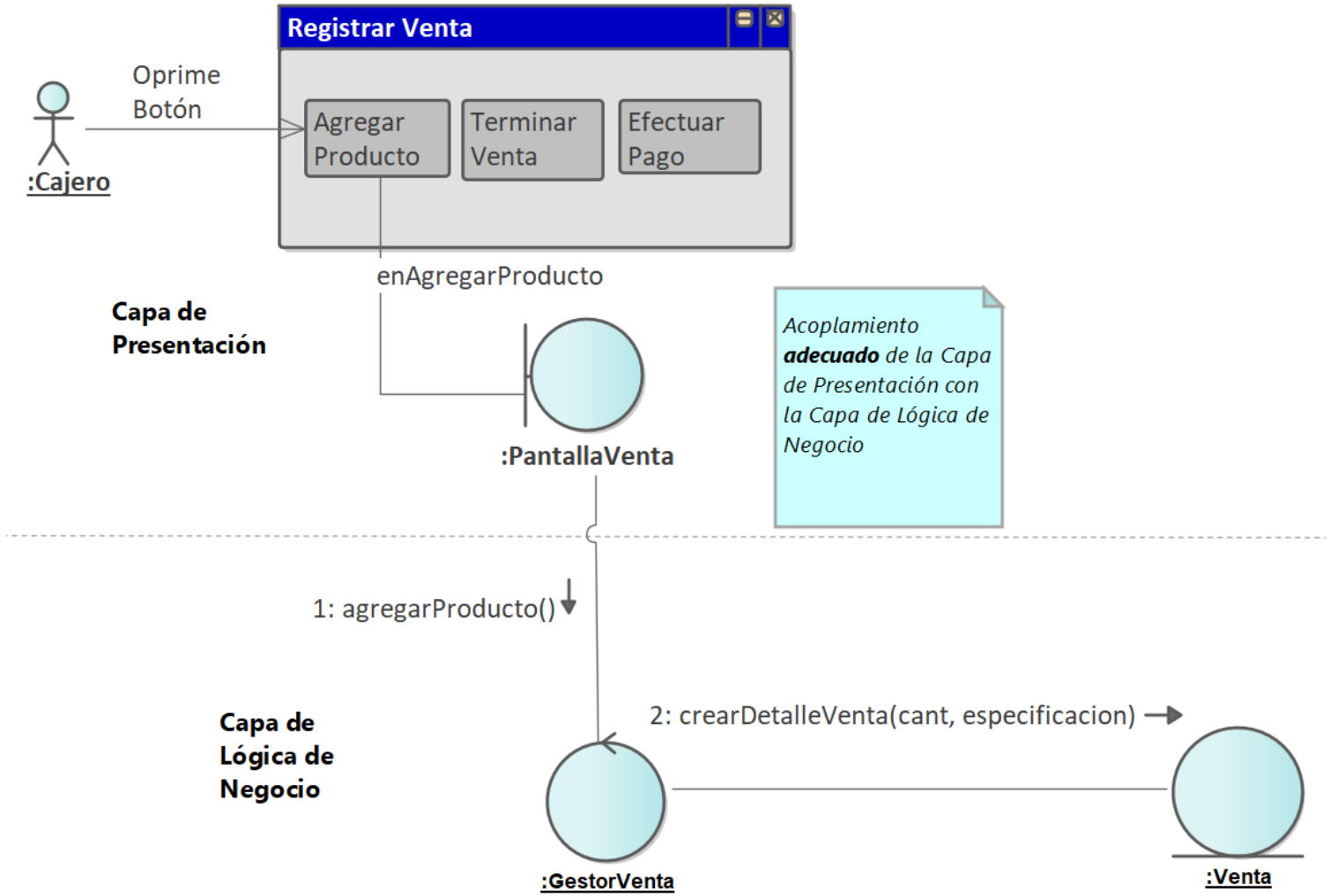


Clase de control modela funcionalidad que operar sobre varios objetos diferentes de entidad, haciendo algunos cálculos y retornando el resultado al objeto de boundary.
Contiene comportamiento de la *lógica de negocio definida en un caso de uso.*
Tiene responsabilidades de *coordinación* de la ejecución de un caso de uso y funciona como *intermediario* entre las clases de interfaz y las de entidad.

Sin aplicar el patrón Controlador

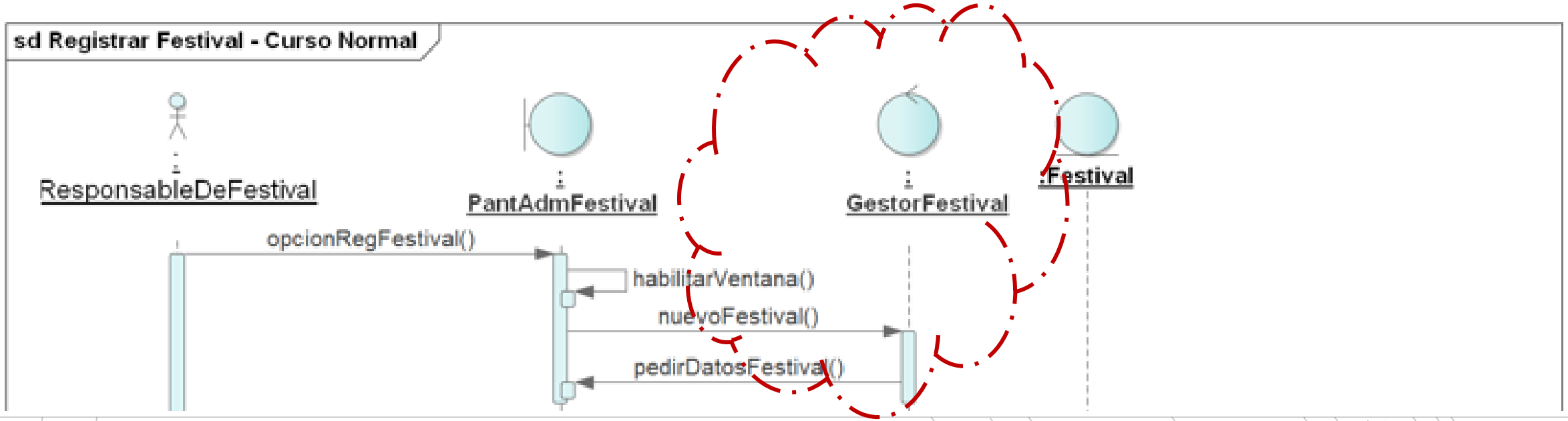


Aplicando el Patrón Controlador



Caso de Estudio: Festival de Folklore – CU Registrar Festival

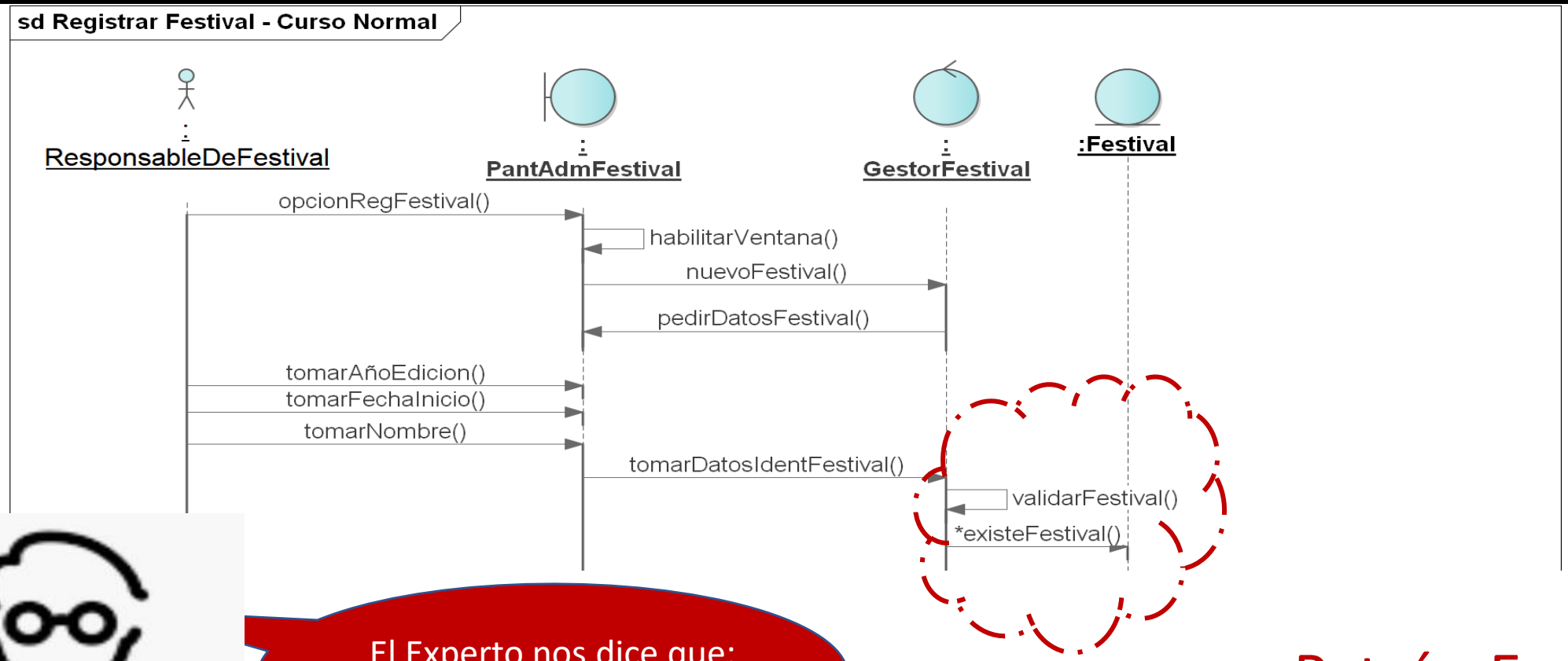
Nombre del Use Case: **Registrar Festival**				ID: 05
Prioridad:	☒ Alta	☐ Media	☐ Baja	
Complejidad:	☐ Simple (1)	☐ Mediano (3)	☒ Complejo (5)	☐ Muy Complejo (8)	☐ Extremadamente Complejo (13)
Actor Principal: Responsable Festival (RF)			Actor Secundario: no aplica	
Tipo de Use Case:	☒ Concreto	☐ Abstracto		
Objetivo: Registrar los datos de un festival y los días que durará.				
Precondiciones: no aplica				
Flujo Básico (Curso Normal)				
1. **RF**: selecciona opción **Registrar Festival**				
2. **SI**: solicita se ingresen los siguientes datos para el festival: año de edición, fecha de inicio, nombre				
3. **RF:** ingresa los datos				
4. SI: valida que no haya registrado en el sistema un festival con el mismo nombre y fecha de inicio, para el año de edición ingresado y no hay.				
5. **SI**: solicita se ingresen los demás datos para el festival: descuento por venta anticipada y porcentaje de devolución por anulación				
6. **RF:** ingresa los datos				
7. **SI**: **para cada día que integrará el festival** se solicita se ingrese: fecha, hora de presentación, fecha límite para anulación de entrada y fecha de vencimiento de venta anticipada				
8. **RF**: ingresa los datos requeridos				
9. **SI**: valida que no haya superposición de fechas para los días del festival ingresados y no hay.				
10. **SI:** solicita confirmación para registrar el festival con sus días				
11. **RF:** confirma la registración.				
12. SI: registra el festival como no vigente y registra cada uno de los días del festival.				
Flujos Alternativos				
A1. Existe festival registrado con los mismos datos. **A2.** Existe superposición de fecha de días de festival. **A3.** NO se confirma registración del festival.				
Observaciones: **1.** RF puede cancelar la operación en cualquier momento seleccionado la opción correspondiente.				



Estructuramos la realización del caso de uso con un Controlador de caso

Patrón Controlador

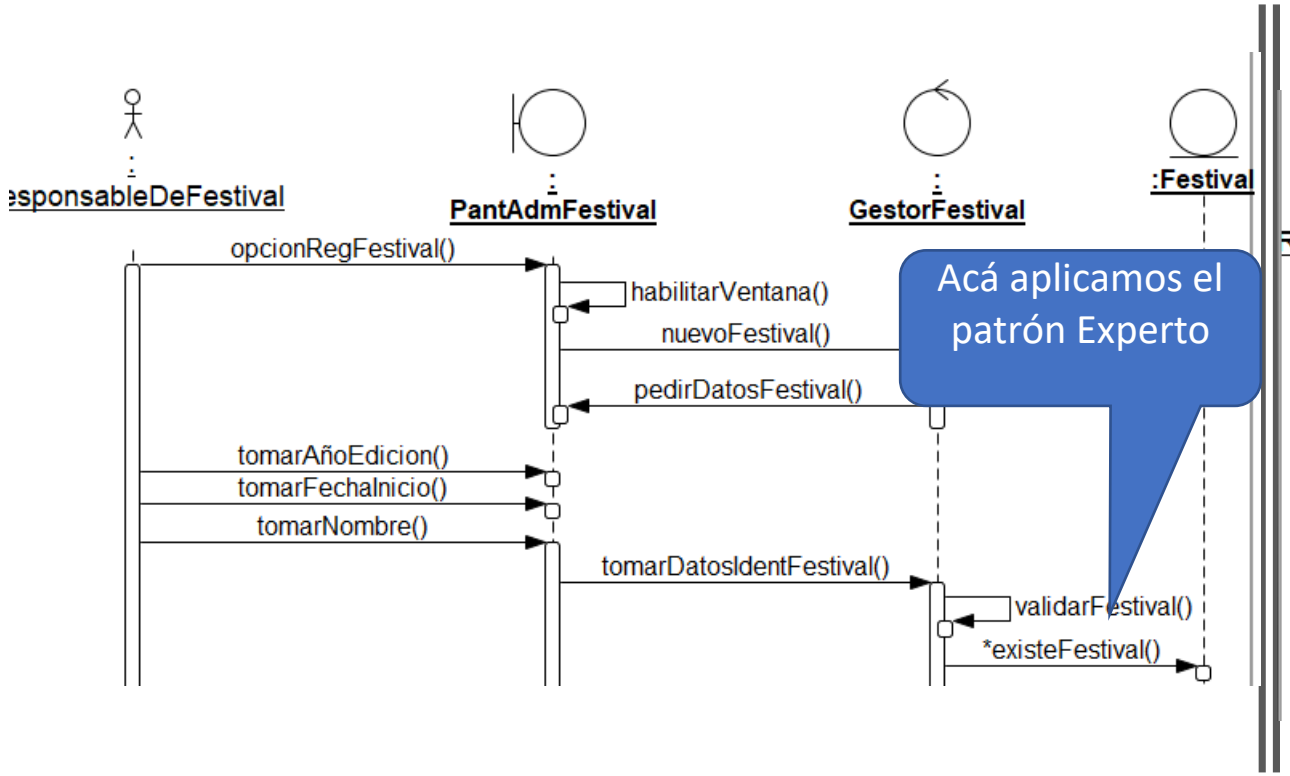
Necesitamos validar si existe unFestival con esos datos, ya registrado en el sistema…

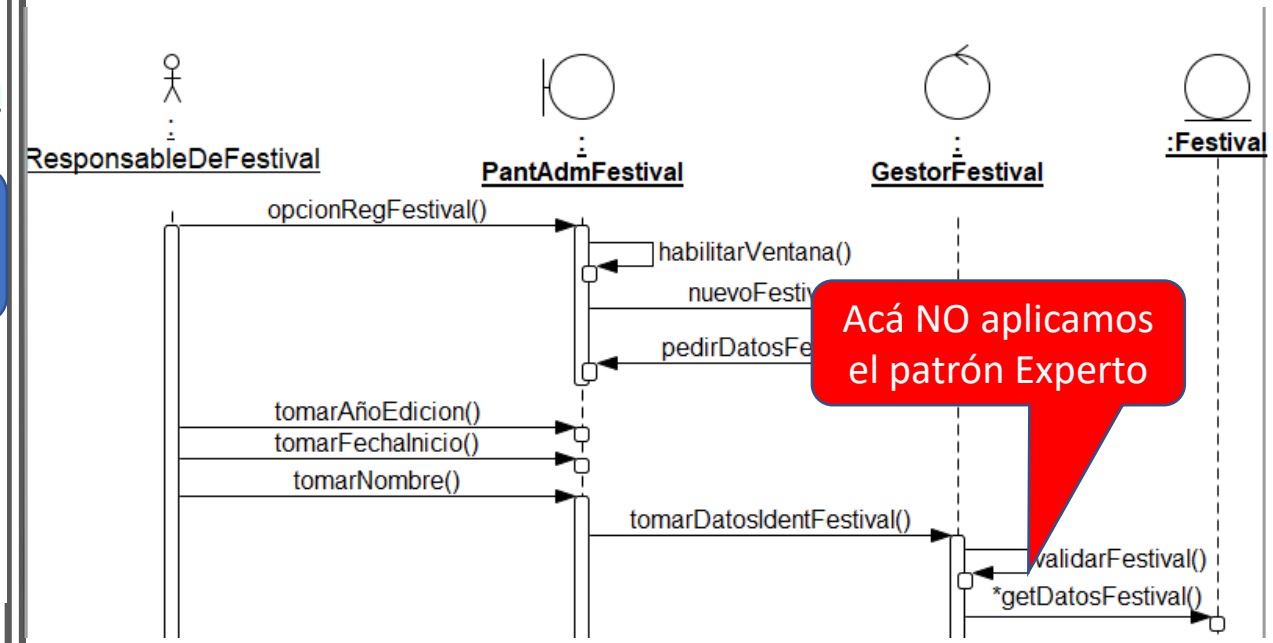


El Experto nos dice que:
"Lo hace quién conoce"

Patrón Experto

¿Qué clase tiene la Información para validar si hay un festival registrado con esos datos?





Ahora necesitamos crear una nueva instancia de Festival, ¿quién tiene la responsabilidad de hacerlo?

Patrón Creador

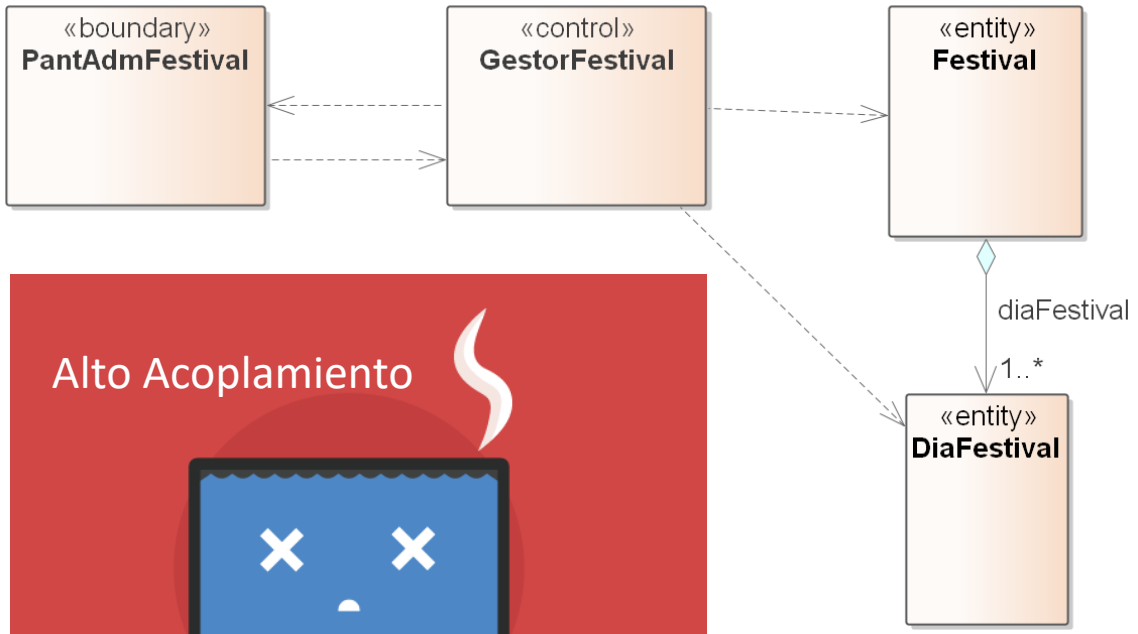
Patrón Creador

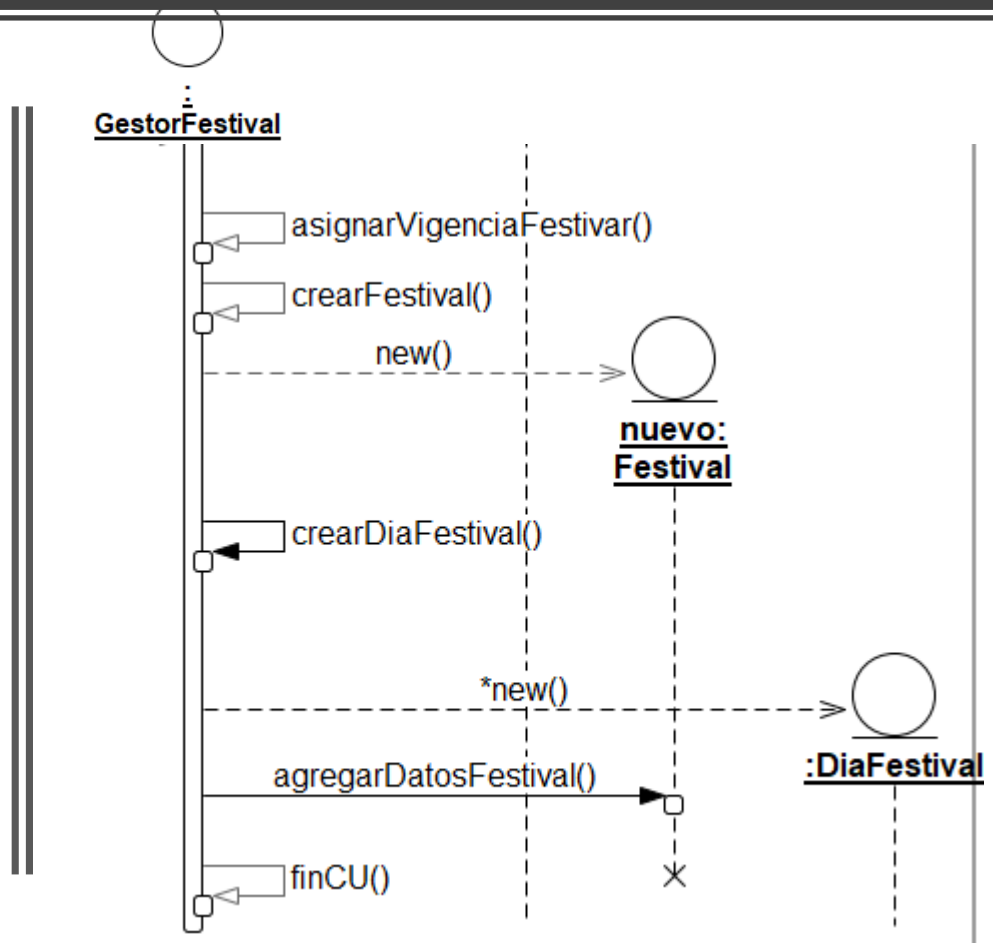
Asignar a la clase B la responsabilidad de crear una instancia de la clase A en uno de los siguientes casos:

* B agrega los objetos de A.
* B contiene los objetos de A.
* B tiene los datos de inicialización que serán transmitidos a A cuando sea creado (así B es un experto respecto de la creación de A).

Ahora bien, cómo se vería si aplicáramos en los dos casos el patrón Creador con la opción:

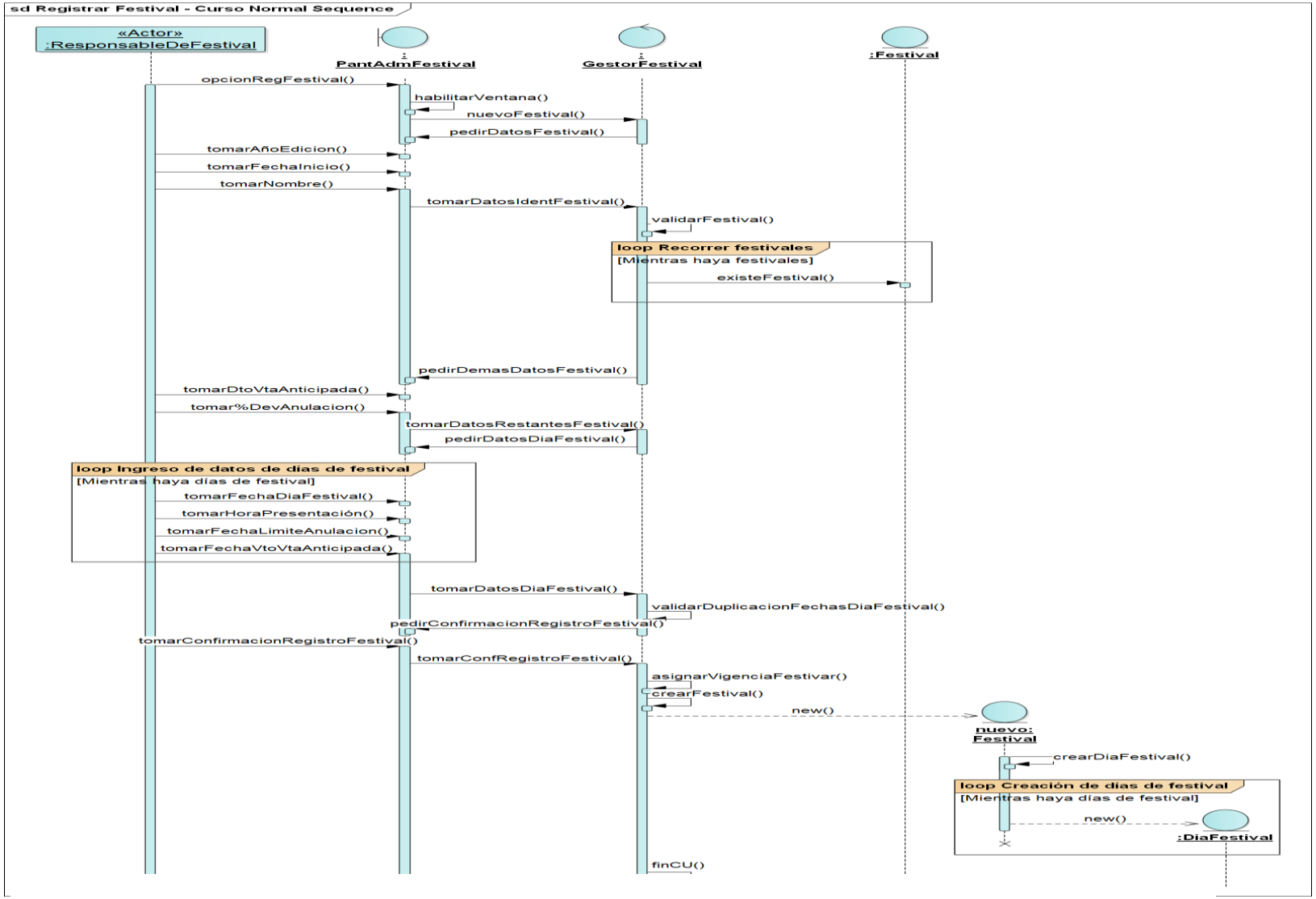
*** B tiene los datos de inicialización que serán transmitidos a A cuando sea creado (así B es un experto respecto de la creación de A).**

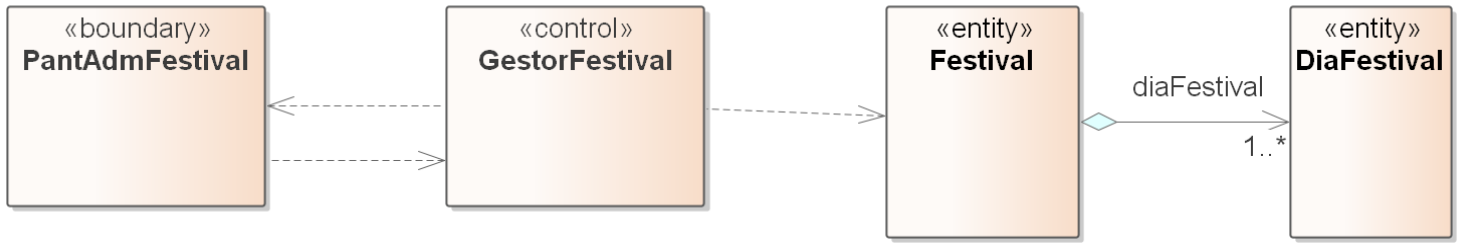


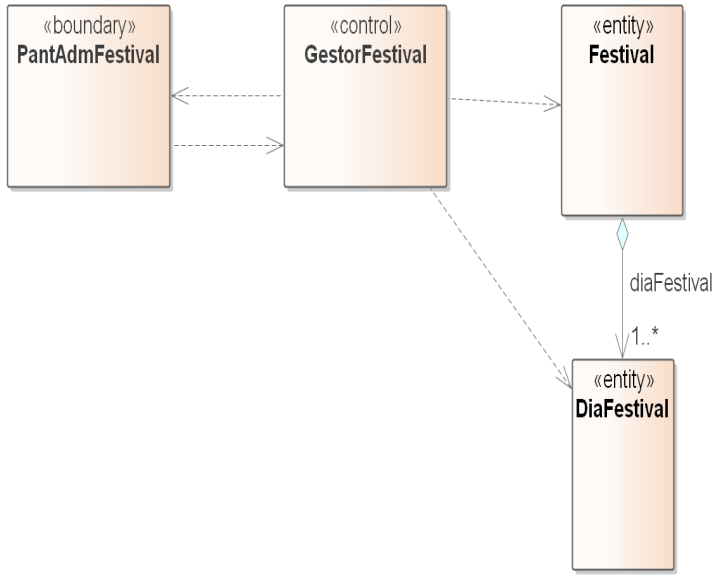


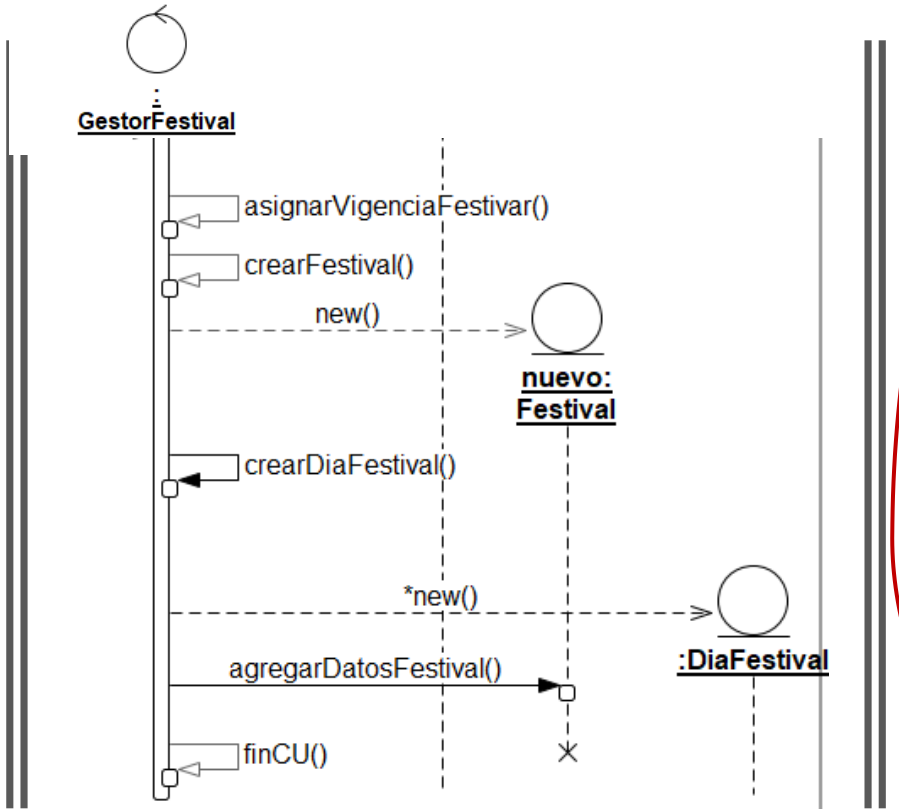
Pasando en limpio…
Esta solución aplica los patrones:

- Creador
- Bajo Acoplamiento









«control»
GestorFestival

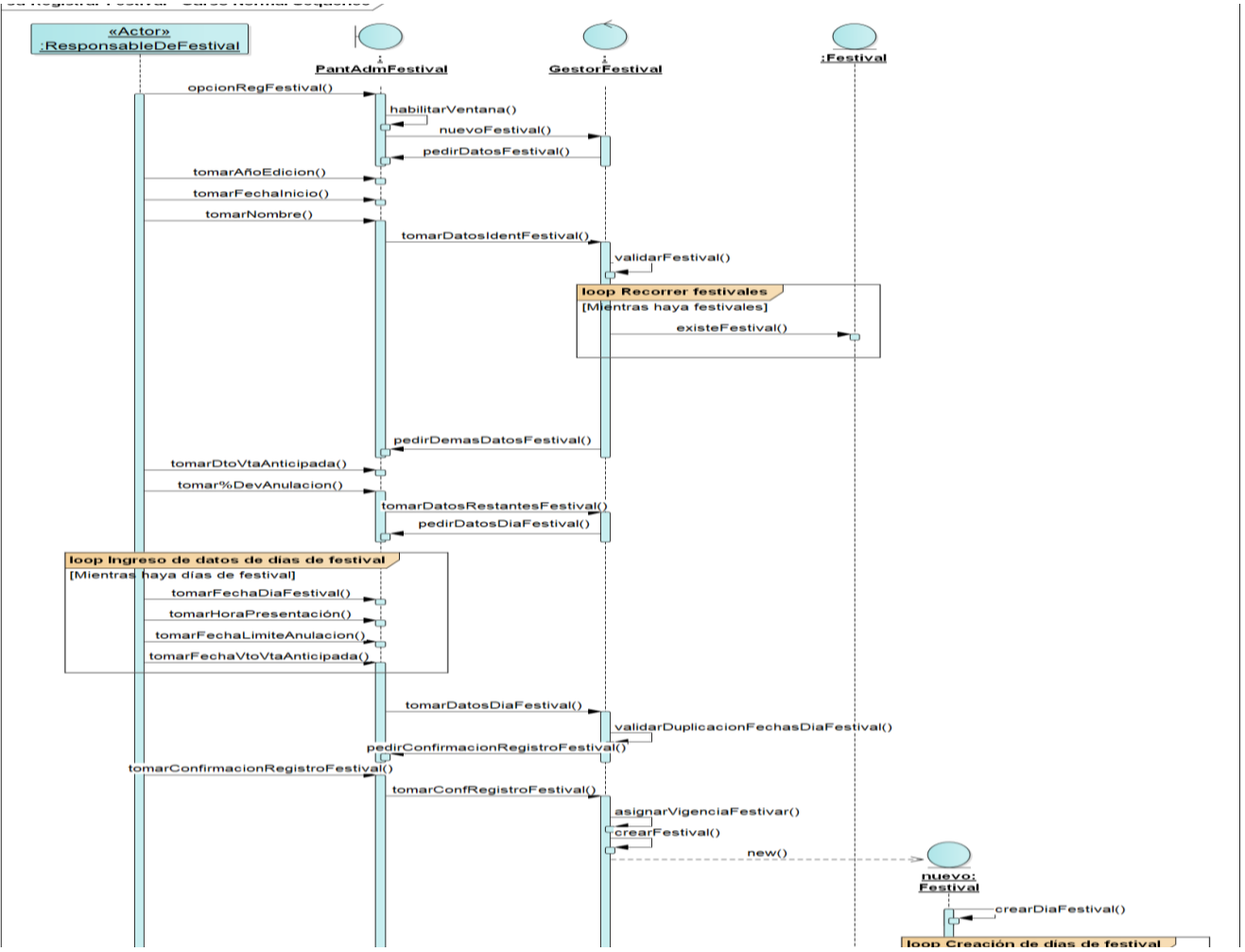
crearDiaFestival()
crearFestival()
finCU()
nuevoFestival()
tomarConfRegistroFestival()
tomarDatosDiaFestival()
tomarDatosIdentFestival()
tomarDatosRestantesFestival()
validarDuplicacionFechasDiaFestival()
validarFestival()

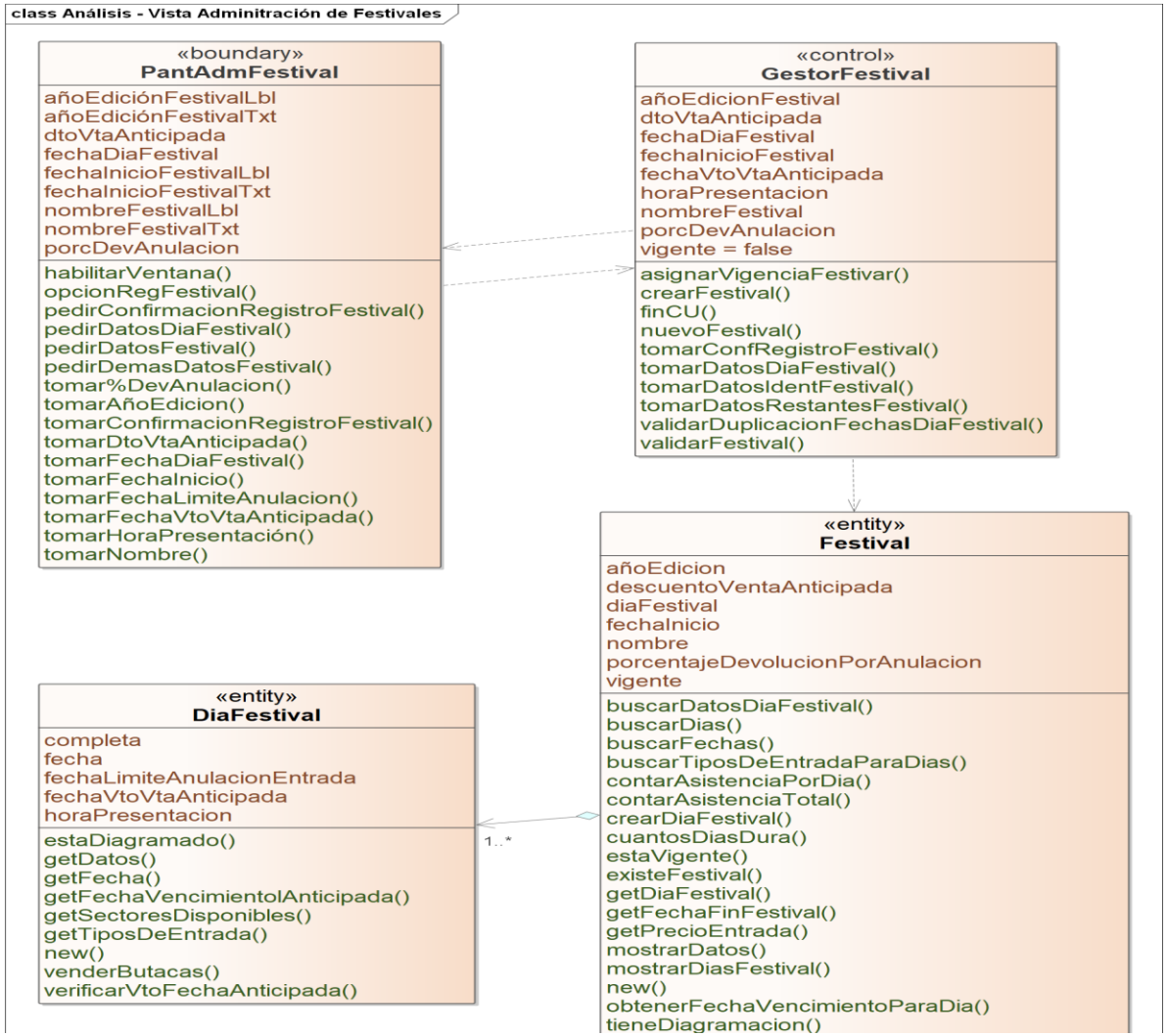
Mientras tanto…

¿Cómo se ve la clase GestorFestival con esta solución?

Entonces la solución elegida aplica los patrones:

- Controlador
- Experto
- Creador
- Bajo Acoplamiento
- Alta Cohesión





Patrón Controlador

Problema

¿Quién debería encargarse de atender un evento del sistema?

Un evento del sistema es un **evento de alto nivel generado por el actor externo.**

Solución

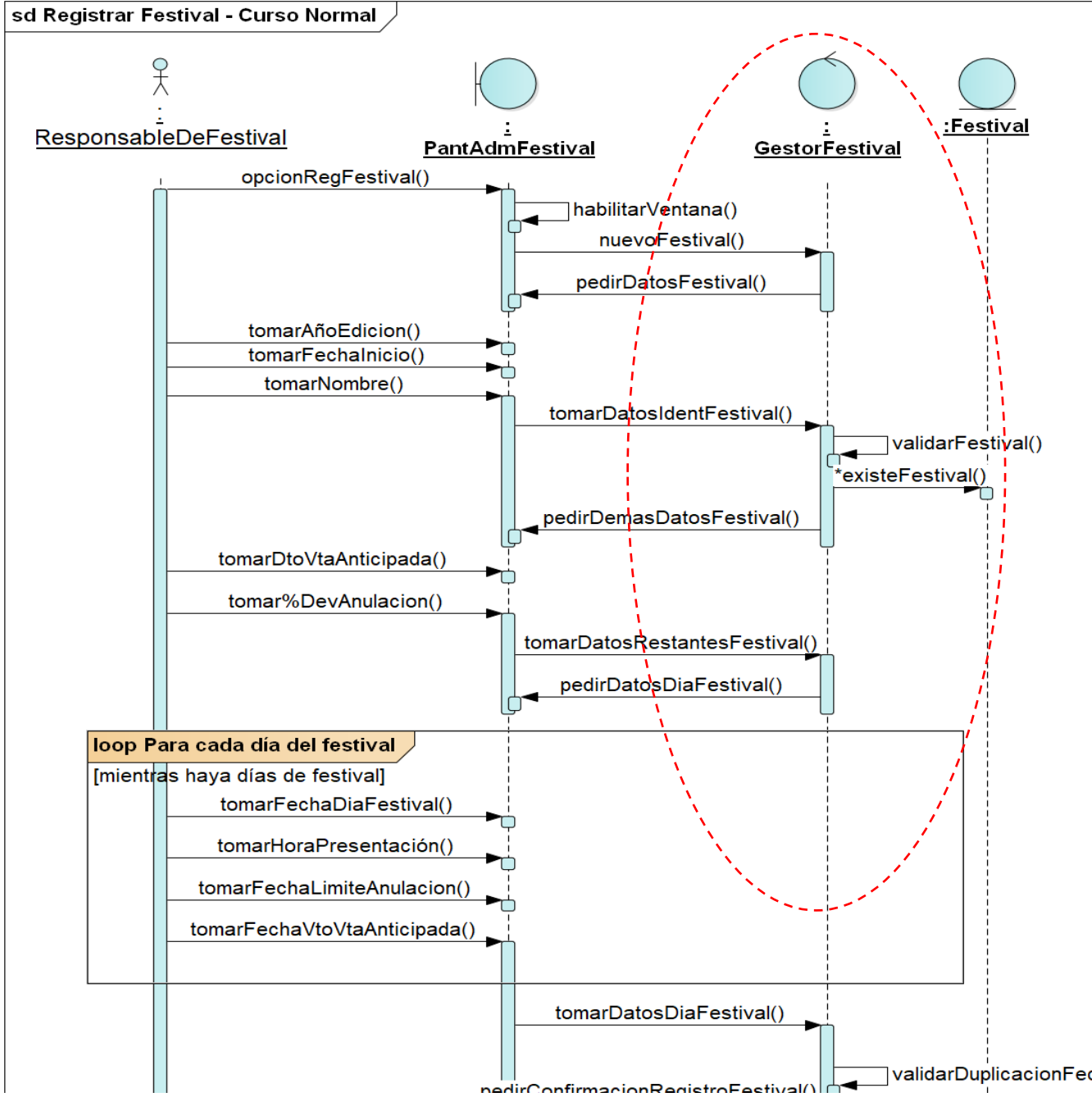
Asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema, a una clase controladora de esos eventos

Explicación

Un controlador de caso de uso es una buena alternativa cuando **hay muchos eventos del sistema entre varios procesos:** asigna su manejo a clases individuales controlables

Beneficios

Garantiza que los procesos del dominio sean manejados por la capa del dominio y no por la de interfaz. Reflexionar sobre el estado de un caso de uso: asegurar que las operaciones ocurran en una cierta secuencia o saber el estado actual de las operaciones del caso de uso.



Patrón Experto

Problema

¿Cuál es el principio fundamental en virtud del cual se asignan las responsabilidades en el diseño orientado a objetos?

Solución

Asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir con la responsabilidad.

Explicación

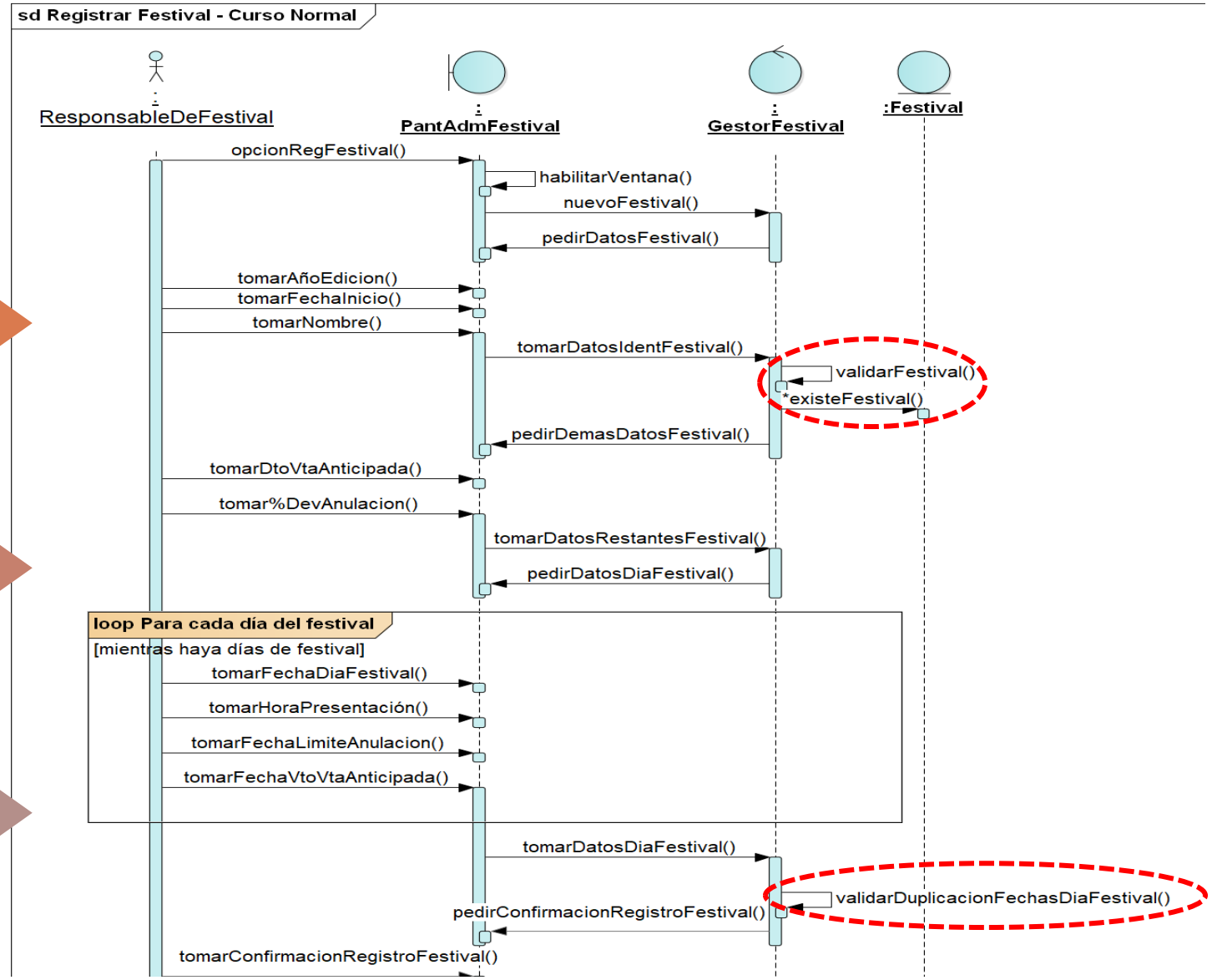
Los objetos hacen cosas relacionadas con la información que poseen.

Puede haber expertos parciales

Beneficios

Bajo el acoplamiento

El comportamiento se distribuye entre las clases que cuentan con la información requerida



Patrón Creador

Problema

¿Quién debería ser el responsable de crear una nueva instancia de alguna clase?

Solución

Asignar a la clase B la responsabilidad de crear una instancia de la clase A cuando:

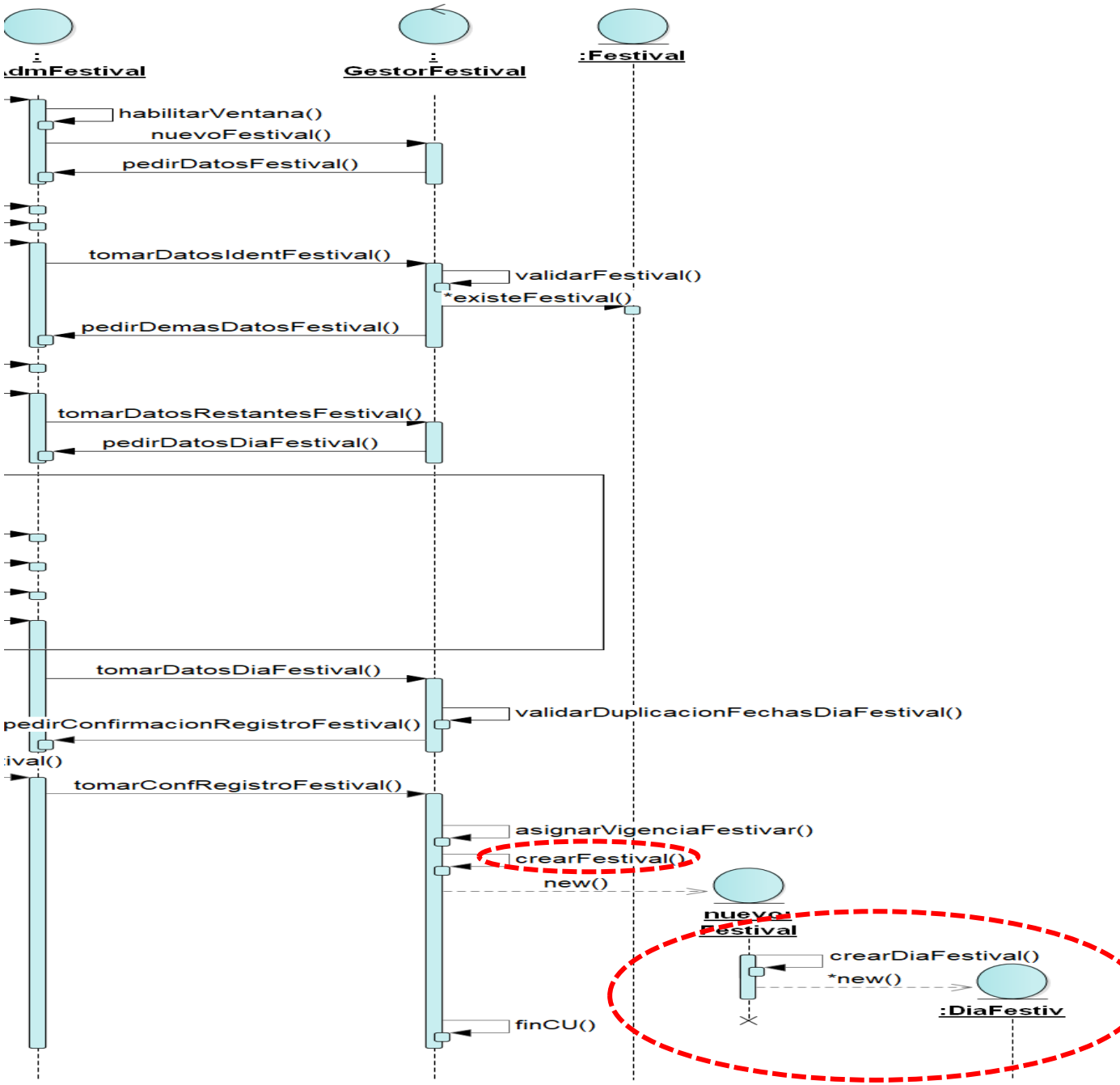
B agrega/ contiene los objetos de A; B utiliza específicamente los objetos de A; B tiene los datos de inicialización que serán transmitidos a A cuando sea creado

Explicación

Encontrar un creador que debemos conectar con el objeto producido, esto soporta el bajo acoplamiento. El agregado “agrega” la parte, el contenedor “contiene” el contenido, el registro “registra”.

Beneficios

Ayuda al bajo acoplamiento, supone menos dependencias. Como la clase creada tiende a ser visible al creador (razón por la cual se la eligió).



Patrón Bajo Acoplamiento

Problema

¿Cómo dar soporte a una dependencia escasa y a un aumento de reutilización?

Solución

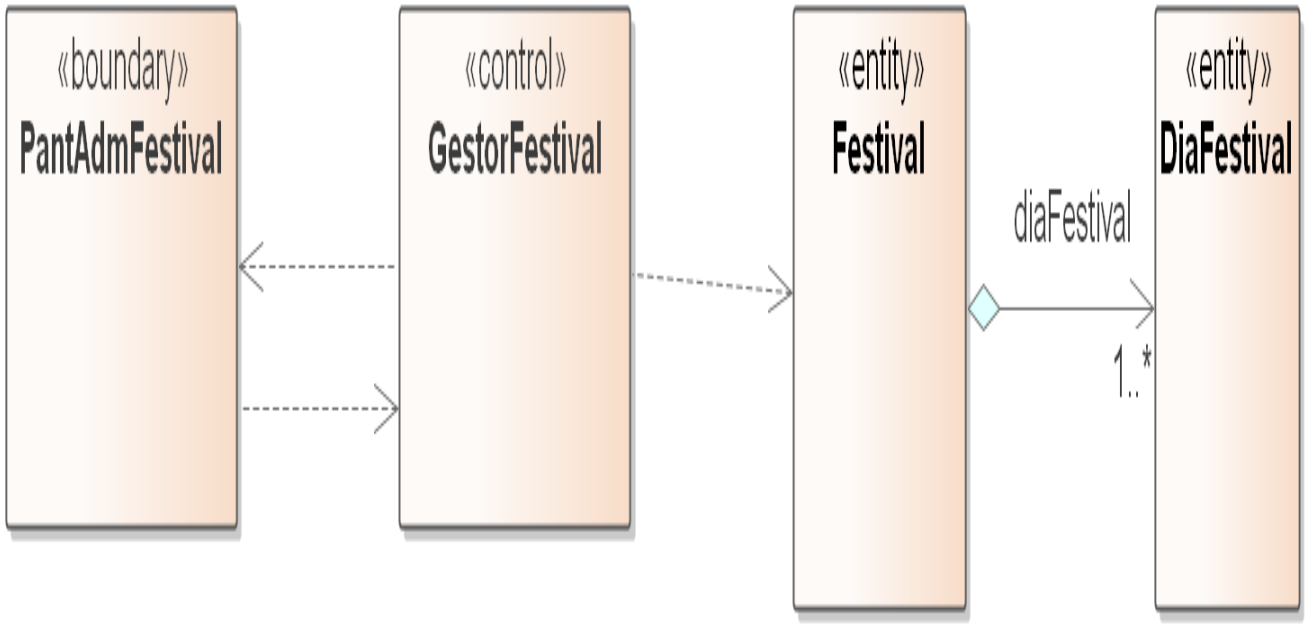
Asignar una responsabilidad para mantener bajo el acoplamiento.

Explicación

Clases más independientes, que reducen el impacto de los cambios, y también más reutilizables. Un grado moderado de acoplamiento es normal para crear un sistema OO, donde los objetos colaboran entre sí.

Beneficios

No se afectan componentes por cambios de otros componentes. Fáciles de entender por separado. Fáciles de reutilizar.



Patrón Alta Cohesión

Problema

Las clases con baja cohesión a menudo representan un grado de abstracción alto o han asumido responsabilidades que deberían haber delegado en otros objetos.

Solución

Asignar una responsabilidad de modo que la cohesión siga siendo alta.

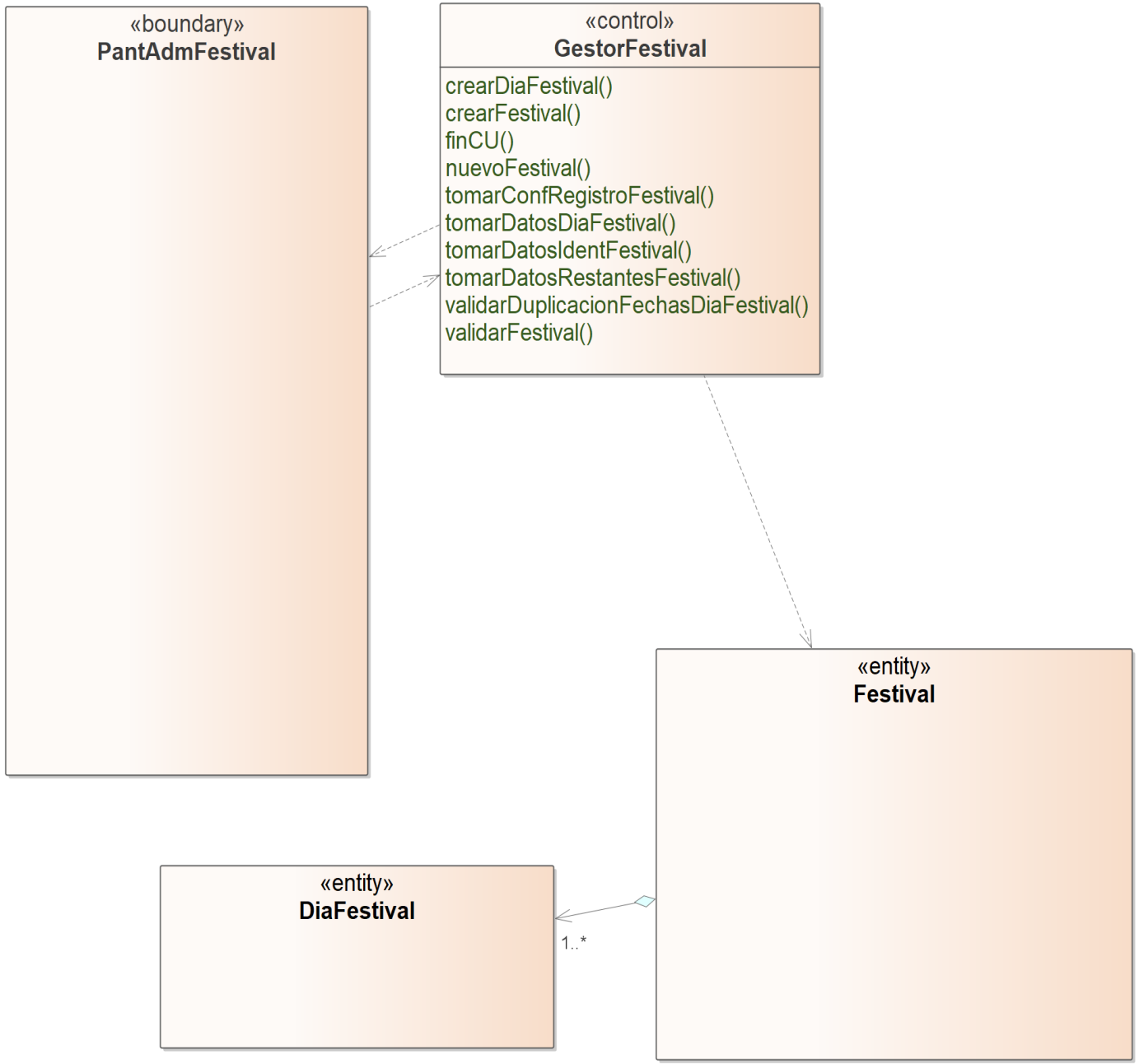
Explicación

Una clase de alta cohesión tiene un número relativamente pequeño de operaciones con funcionalidad relacionada y poco trabajo por hacer. Colabora con otros objetos para compartir esfuerzo si la tarea es grande.

Beneficios

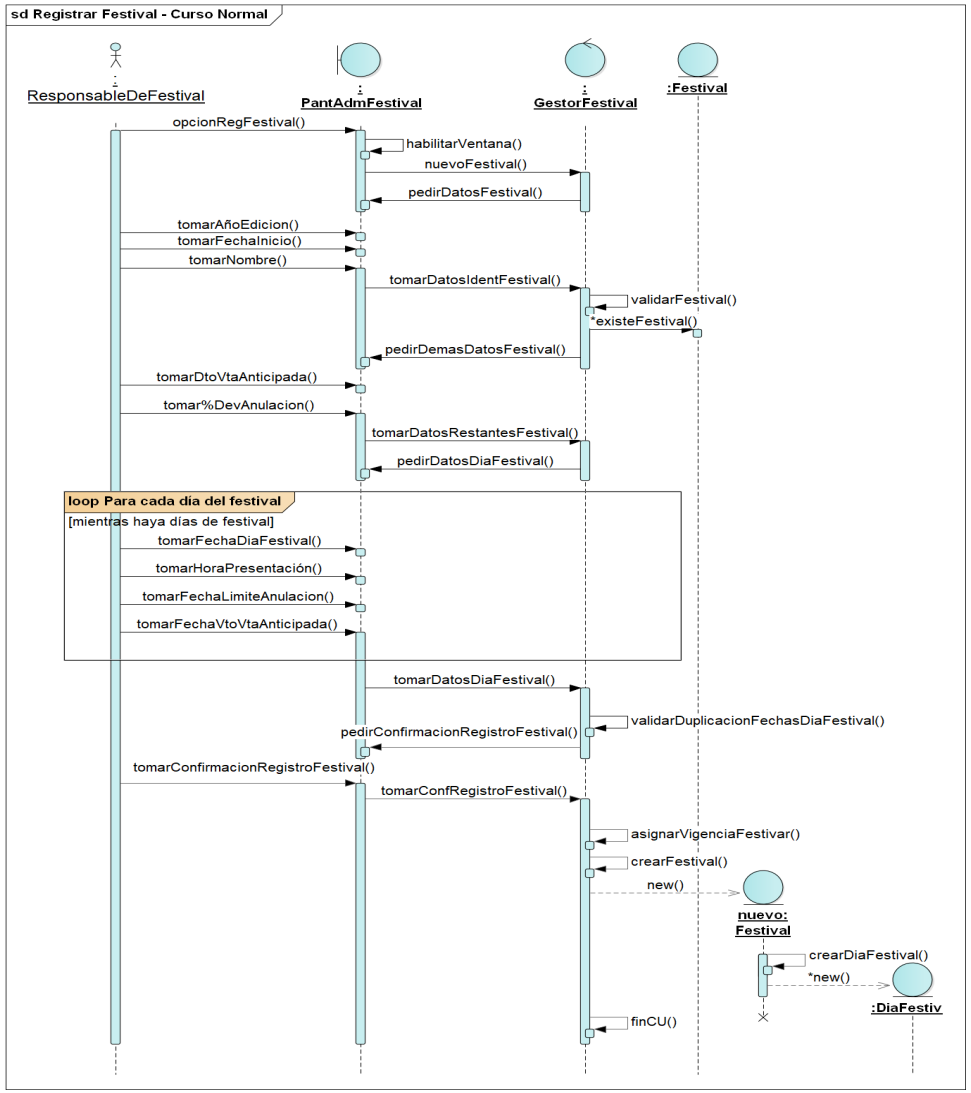
Mejoran la claridad y la facilidad de comprensión Simplifican la evolución.

A menudo se genera el bajo acoplamiento. Soporta una mayor capacidad de reutilización, porque una clase muy cohesiva puede destinarse a un propósito muy específico

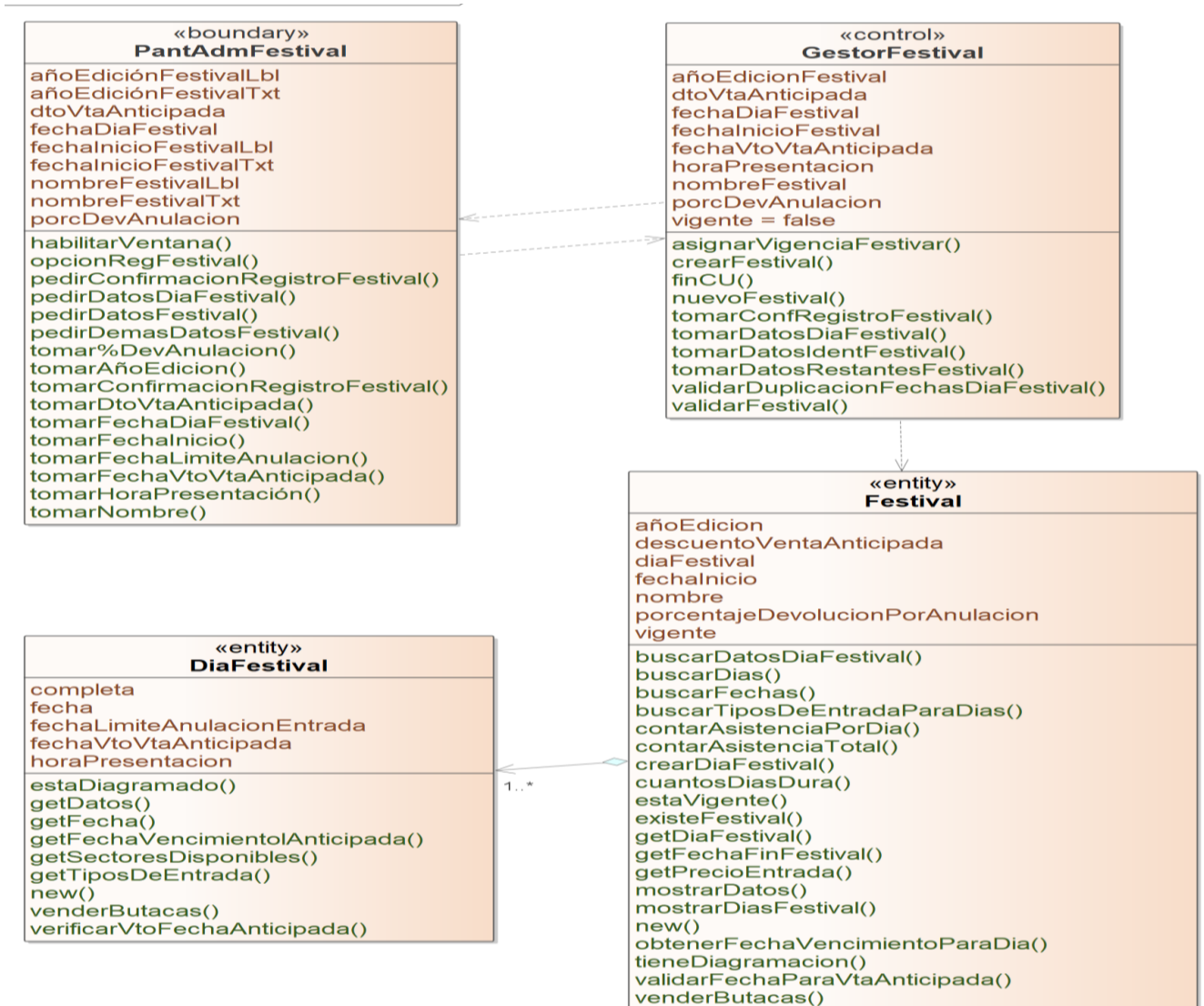


Realización de Análisis del Caso de Uso Registrar Festival

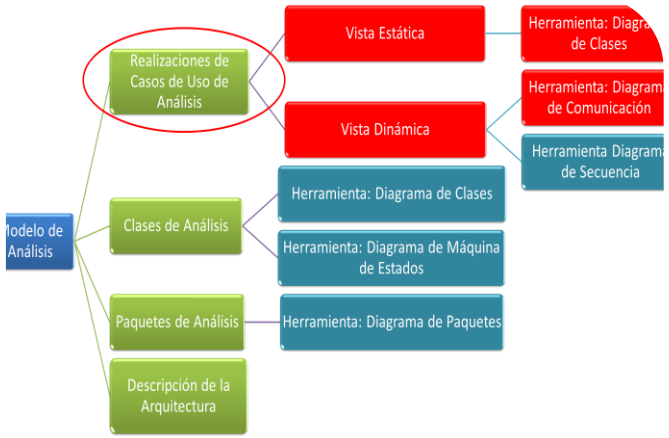
Vista Dinámica con Diagrama de secuencia para el escenario del Curso Normal



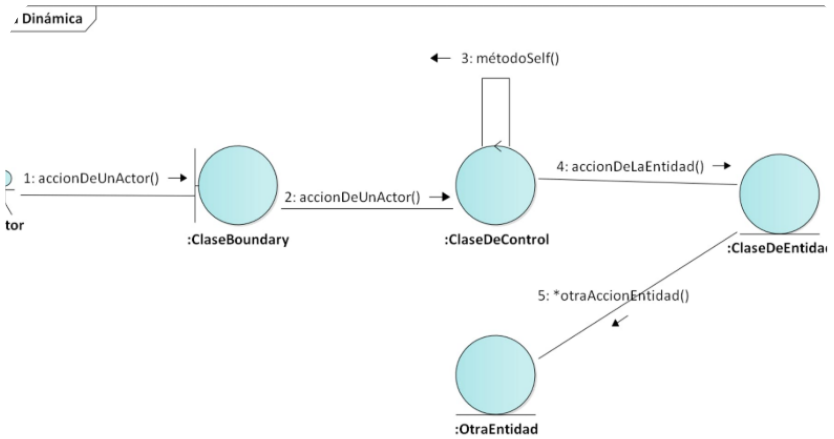
Vista de Estructura de Clases de análisis con Diagrama de Clases



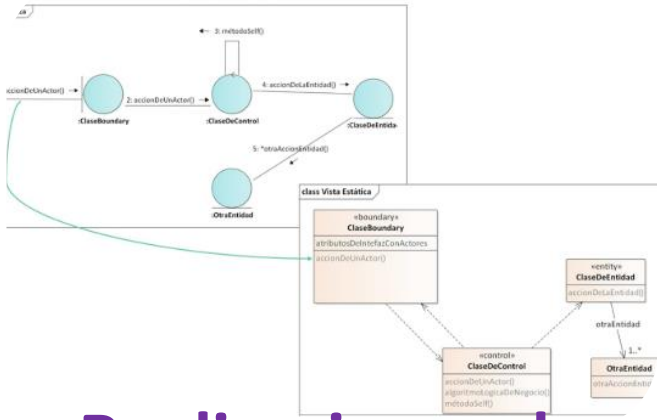
Repasando lo que vimos…



Modelo de Análisis



Clases de Análisis



Realizaciones de CU de Análisis

Patrones GRASP para Análisis

Patrones Grasp para Análisis

- Experto en Información
- Creador
- Bajo Acoplamiento
- Alta Cohesión
- Controlador

Finalmente…

Muchas Gracias!!!