Análisis y Diseño de Sistemas

Tema: CRC

Urciuolo A.

UNPSJB – 2015

El Proceso de AyD CRC

- Es un proceso por el cual los requerimientos del software son convertidos en especificaciones detalladas de los objetos.

- La especificación incluye una descripción completa de los respectivos roles y responsabilidades de los objetos y cómo se comunican entre ellos.

CRC. La exploración inicial

- Al comienzo, el proceso del DOO es exploratorio. El diseñador busca clases, probando diversos esquemas, en orden a descubrir el modo más natural de modelar el sistema.

- El proceso de AOO inicialmente consiste de los siguientes pasos:
- Encontrar las clases en el sistema.
- Determinar qué operaciones es responsable cada clase de ejecutar y qué conocimiento debe mantener.
- Determinar la forma en la cual objetos colaboran con otros objetos en orden a descargar sus responsabilidades.

CRC. La exploración inicial

Estos pasos producen:

- **Una lista de las clases dentro de la aplicación.**
- **Una descripción del conocimiento y operaciones para las cuales cada clase es responsable.**
- **Una descripción de las colaboraciones entre clases.**

CRC. El Diseño detallado

- Con la información obtenida, se comienza un paso más analítico del proceso de diseño:
  - Factorizar responsabilidades comunes en orden a construir jerarquías de clases.
  - Analizar las colaboraciones entre clases para simplificarlas

CRC. Subsistemas de clases

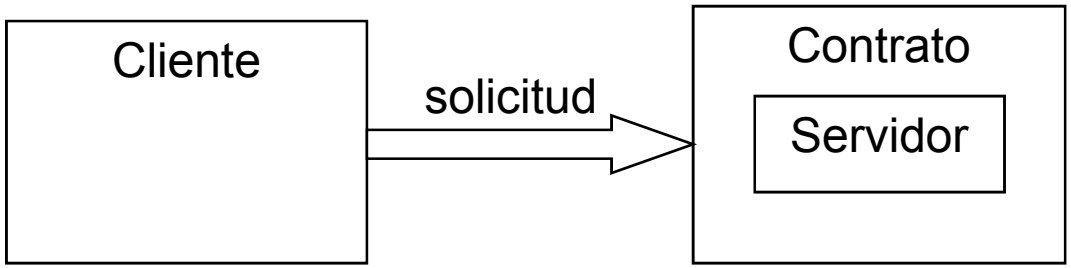
- Las clases son una forma de particionar y estructurar una aplicación para reuso.
- También pueden identificarse piezas que tienen cierta integridad lógica, pero que pueden descomponerse en piezas más pequeñas, referidas como subsistemas.
- Un subsistema es un conjunto de clases (y posiblemente otros subsistemas), colaborando para alcanzar un conjunto de responsabilidades.

Clientes y servidores

- Las colaboraciones entre objetos son vistas como interacciones en un sentido:
  - Un objeto requiere un servicio de otro objeto
  - El objeto que hace la solicitud es el cliente.
  - El objeto que recibe la solicitud y en consecuencia provee el servicio, es el servidor.

- Los modos en los cuales un cliente dado, puede interactuar con un servidor son descriptos por un contrato.
- Un contrato es una lista de solicitudes que un cliente puede hacer a un servidor.

Clientes y servidores



Contrato Cliente-Servidor

Se modelan los sistemas como clientes y servidores que colaboran en modos especificados por contratos.

Encontrar los objetos

- Este es un problema de modelado. El modelado es el proceso por el cual, los objetos lógicos del problema (del mundo real) son mapeados a los objetos actuales del programa.

- Se determinan las metas del diseño.
- Se determina qué comportamiento debe ser claramente llevado a cabo por el sistema.
- Se define el sistema dibujando fronteras y determinando qué está adentro y qué está afuera del sistema.
- Para cada meta del sistema, se debe preguntar: qué tipo de objetos son necesarios para alcanzarla?

Determinar sus responsabilidades

- Cuando se ha hecho una aproximación de listado de los objetos requeridos, entonces se puede preguntar qué objeto individual lo realizará.

  - Qué tiene que conocer cada objeto para alcanzar cada meta en la que está involucrado?
  - Qué pasos es responsable de dar para alcanzar cada meta de la cual es responsable?

Determinar colaboraciones

- Con quién colaborará cada objeto en orden a alcanzar cada una de sus responsabilidades?
- Qué otros objetos en el sistema tienen la información que este necesita o conocen cómo ejecutar algunas operaciones que requiere?
- Cuál es la naturaleza de las colaboraciones entre objetos?

Al final del proceso se tendrá:

- Una lista de entidades dentro de la aplicación para jugar el rol de clientes y servidores.
- Una lista de las colaboraciones entre ellos para servir como base de los contratos.

CRC

Clases – Responsabilidades- Colaboraciones

- Encontrar los conceptos centrales del problema
- Definir las responsabilidades de cada concepto
- Encontrar las colaboraciones entre los conceptos

Clases

- La primera meta del diseño es crear clases de objetos para modelar el dominio de la aplicación.

Cómo se identifican las clases?

- El punto de partida es la especificación de requerimientos.
- Leer ciudadosamente la especificación, buscando sustantivos. Cambiar plural a singular.
- Hacer una lista preliminar.
- Dividir en tres categorías: clases obvias, clases sin sentido y frases de las cuales no se está seguro.
- Descartar las clases sin sentido.
- De las otras dos categorías, recoger clases candidatas.

Clases – Tener en cuenta…

- Si se puede poner nombre a una abstracción, se encontró una clase candidata.
- Si se puede formular una declaración de propósitos para esa clase candidata, las chances son mayores.
- Modelar objetos físicos y entidades conceptuales.
- Si existe más de una palabra para un concepto, elegir la más significativa para el resto del sistema.
- Ser ciudadoso con los adjetivos. Pueden significar un tipo diferente de objeto, un uso diferente del mismo objeto, etc.
- Ser ciudadoso con aquellas clases cuyos objetos no son parte del sistema (por ej, “el usuario”)
- Modelar categorías de clases, en esta instancia, como clases específicas, individuales.

Registrar clases candidatas

- El resultado de este procedimiento es la primera y tentativa lista de clases en el programa. Algunas serán olvidadas y otras, eliminadas después..

- Cuando se han identificado las clases candidatas, **escribir su nombre, en una tarjeta para cada una**.
- En la parte posterior de la tarjeta, escribir brevemente una descripción del propósito general de la clase.

Encontrar clases abstractas

- Una vez definida una lista de clases candidatas, ésta se reexamina para encontrar la mayor cantidad de clases abstractas posible.

- *Una clase abstracta surge de un conjunto de clases que comparten un atributo que implica un comportamiento compartido para las clases que tienen ese atributo*.

- Buscar atributos comunes en las clases, según están descriptas en los procedimientos.

Identificar Superclases

- Identificar candidatos para superclases abstractas agrupando clases relacionadas. Una clase puede aparacer en más de un grupo.
- Una vez identificado un grupo, nombre la superclase que se piensa que representa. Usar un nombre singular.

Por ej:

- Display Screen
- Printer
- Floppy Disk
- Mouse

- Su atributo compartido es que todas estas clases son interfaces a dispositivos físicos. Se puede nombrar este grupo como Dispositivo Físico.

Registrar Superclases

- Cuando se han identificado las superclases, se registran en tarjetas y se escriben las subclases en las líneas bajo el nombre.
- Volver atrás a las tarjetas de las clases y registrar sus superclases y subclases, si ya se conocen en las líneas bajo sus nombres.

Responsabilidades

- Incluyen dos items clave:
  - el conocimiento que un objeto mantiene
  - las acciones que el objeto ejecuta.

- Las responsabilidades de un objeto son todos los servicios que provee para todos los contratos que soporta.

- En esta etapa del diseño, se consideran las responsabilidades generales, no específicas, tratando de mantener todas las responsabilidades de una clase en el mismo nivel conceptual.

Responsabilidades

- En el comienzo del diseño, las fuentes son: la especificación de requerimientos y las clases que ya se han identificado.

- Leer nuevamente la especificación de requerimientos.
- Resaltar todos los verbos y usar el criterio para determinar cuáles de estos representan claramente acciones que algún objeto dentro del sistema debe ejecutar.
- Cada vez que se mencione información, resaltarla.
- Realizar un walk-through, imaginando cómo sucederían las invocaciones en el sistema y yendo a través de una variedad de escenarios usando tantas capacidades del sistema como sea posible.

Responsabilidades

- Usar las clases identificadas. El nombre elegido para la clase sugiere la responsabilidad primaria de la clase y posiblemente otras.
- La declaración de propósitos de la clase, también puede llevar responsabilidades adicionales. Qué conocimiento y acciones están implicadas en este propósito?
- Comparar y contrastar los roles de las distintas clases, también puede generar nuevas responsabilidades.

Asignando Responsabilidades

- Asignar cada responsabilidad identificada a la o las clases a las que lógicamente pertenezca.

- Seguir los siguientes lineamientos:
  - Distribuir equitativamente la inteligencia del sistema.
  - Asignar responsabilidades lo más generalmente posible.
  - Mantener el comportamiento con la información relacionada.
  - Mantener toda la información sobre una cosa, en un lugar.
  - Compartir responsabilidades entre objetos relacionados.

Registrar Responsabilidades

- En cada tarjeta de clase creada, se debe registrar cada responsabilidad asignada a la misma.
- Listar las responsabilidades lo más suscintamente posible, una frase para cada una.
- Una clase no debe estar sobrecargada de responsabilidades. Puede ser una señal de haber señalizado la inteligencia del sistema en alguna clase.
- Si la responsabilidad ya fue listada en la tarjeta de la superclase, no necesita listarse en la de la subclase.

Colaboraciones

- Las colaboraciones representan las solicitudes de un cliente a un servidor en cumplimiento de la responsabilidad de un cliente.
- Una colaboración representa una materialización del contrato entre el cliente y el servidor.
- Un objeto puede alcanzar una responsabilidad particular por sí mismo o puede requerir la asistencia de otros objetos.
- Un objeto colabora con otro si, para alcanzar su responsabilidad, necesita enviar mensajes a otro objeto.
- Una colaboración simple fluye en una dirección, representando la solicitud del cliente al servidor.
- Cada colaboración del cliente, está asociada con una responsabilidad particular implementada por el servidor.

Colaboraciones

- Aunque cada colaboración, trabaja para alcanzar una responsabilidad, alcanzar una responsabilidad no necesariamente requiere una colaboración.
- Puede requerir muchas colaboraciones, alcanzar completamente una responsabilidad única.
- Algunos objetos, alcanzarán una responsabilidad sin colaborar con ningún otro objeto.
- La razón de la importancia de las colaboraciones es que el patrón de colaboraciones dentro de la aplicación, revela el flujo de control e información durante su ejecución..

Encontrando Colaboraciones

- Comenzar analizando las interacciones de cada clase.
- Examinar las responsabilidades por dependencias.
- Para identificar colaboraciones, realizar las siguientes preguntas para cada responsabilidad de cada clase:
  - Es la clase capaz de alcanzar esta responsabilidad por sí misma?
  - Si no lo es, qué necesita?
  - De qué otra clase puede adquirir lo que necesita?
  - Similarmente, para cada clase, preguntar:
  - Qué hace o conoce esta clase?
  - Qué otra clase necesita los resultados o información?
- Si una clase no tiene interacciones con otras clases, debe ser descartada.

Encontrando Colaboraciones

- Mientras identificamos responsabilidades, examinar las relaciones entre clases puede servir para identificar colaboraciones:
  - Relación “es parte de”
  - Relación “tiene conocimiento de”

Registrar Colaboraciones

- Tomar la tarjeta para la clase que juega el rol de cliente.
- Escribir el nombre de la clase que juega el rol de servidor directamente a la derecha de la responsabilidad que la colaboración sirve para ayudar a alcanzar.
- Si la responsabilidad requiere muchas colaboraciones, escribir el nombre de cada clase requerida para alcanzar la responsabilidad.
- Si muchas responsabilidades, todas requieren una clase para colaborar con las misma otra clase, registrar varias colaboraciones, una para cada responsabilidad.

Registrar Colaboraciones

- Estar seguro de que existe la correspondiente responsabilidad para cada colaboración que se registre.
- Los servicios que provistos por una clase, incluyen aquellos listados en su tarjeta y las responsabilidades que heredan de sus superclases.
- Si para alcanzar una responsabilidad se requiere colaboración con otras instancias de la misma clase (o de sus superclases), registrar también esa colaboración.

Ejemplo Biblioteca: Modelar un sistema de biblioteca con los siguientes requerimientos:

- El sistema soportará operaciones de búsqueda y préstamo de material bibliográfico que incluye libros, proyectos y revistas.
- Los usuarios de la biblioteca podrán realizar consultas sobre el material bibliográfico
- Cada usuario puede tener prestado un número máximo de elementos
- Cada tipo de material tiene diferentes periodos de préstamo (por ejemplo, los libros tienen un máximo de 7 días en tanto que los proyectos no deberán superar 15 días)
- Si se devuelve un elemento después de la fecha prevista, la biblioteca multará al usuario con diferentes montos económicos dependiendo del tipo de material

Clases candidatas:

- SistemaBiblioteca: sus obligaciones serán las de prestar y devolver materiales, y visualizar y obtener información del usuario
- Biblioteca: con ella colaborará el sistema_de_biblioteca para obtener información sobre los diferentes materiales. El objeto BIBLIOTECA deberá colaborar con un subsistema de base de datos para almacenar la información sobre el material

Clases candidatas:

- Clase Socio: deberá conocer los datos del socio (nombre, dirección, teléfono, etc.), además de saber los elementos que tiene prestados.
- Clase Libro: deberá conocer sus datos (ISBN, título, etc.); poder prestarse y devolverse; calcular la fecha de devolución prevista y la multa en caso necesario.
- Clase Proyecto: deberá conocer sus datos (título, autor, etc.). En caso necesario poder prestarse y devolverse, calcular la fecha de devolución prevista y la multa en caso necesario.
- Clase Revista: tendrá que conocer sus datos (título, idioma, etc.), en caso necesario poder prestarse y devolverse; calcular la fecha de devolución prevista y la multa en caso necesario

Tarjetas CRC:

Clase: Nombre de la clase	
Lista de Superclases	
Lista de Subclases	
Responsabilidad	Colaboración

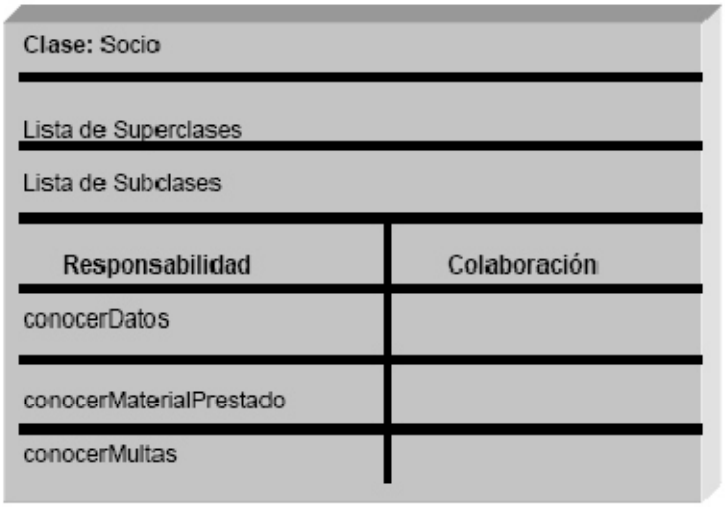
TARJETA DE CLASE

Tarjetas CRC:

Clase: Sistema de Biblioteca	
Lista de Superclases	
Lista de Subclases	
Responsabilidad	**Colaboración**
prestarMaterial	Material, Socio
devolverMaterial	Material, Socio
busquedaMaterial	Biblioteca
visualizarInformacion	
obtenerInformacion	

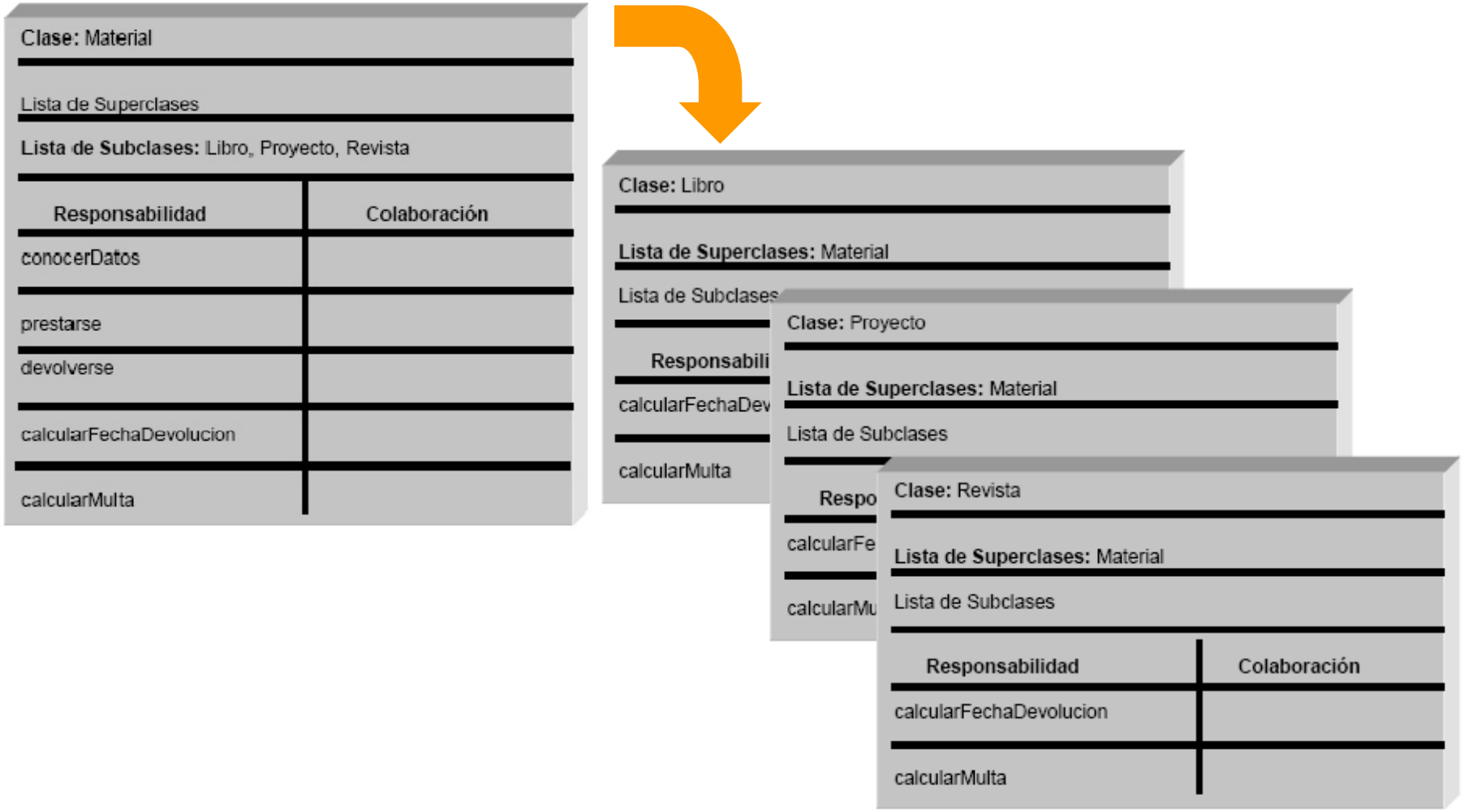
Clase: Biblioteca	
Lista de Superclases	
Lista de Subclases	
Responsabilidad	**Colaboración**
conocerMateriales	Material

Tarjetas CRC:



Tarjetas CRC:

Jerarquía de clases



Funcionamiento del Sistema

- Al prestar un libro, el usuario actúa con el **Sistema_de_biblioteca** (por medio de una interfaz de usuario) y al pedir un libro, el sistema puede colaborar con el objeto **socio** para ver si tiene ya el máximo de elementos prestados y el **sistema_de_biblioteca** emitirá un mensaje colaborando con el subsistema de interfaz.
- En caso de que no supere el máximo, el usuario seleccionará un libro a través del **sistema_de_biblioteca**, que colaborará para ello con la **Biblioteca** y el **libro** que corresponda. El libro se cambiará de estado al prestarse y el objeto **socio** registrará que ese libro lo tiene prestado.
- Así se seguirían validando y mejorando las clases, creando nuevas responsabilidades y colaboraciones e incluso nuevas clases, continuando con el diseño y hasta llegar a la implementación.

Bibliografía

- Wirfs-Brock, R.WilkersonB. *Designing Object-Oriented Software* Prentice Hall, 1990

- Dennis A, Wixom B, and Tegarden D (2005) *System Analysis and Design with UML version 2* second edition, Wiley

  Cap. 7

- Bennett, S., McRobb, S. and Farmer, R. (2002) *Object-Oriented Systems Analysis and Design using UML,* 2nd Edition, McGraw-Hill

  Pag. 168-176