UNA METODOLOGÍA PARA EL DISEÑO DE ESTRUCTURAS DE DATOS COMPLEJAS

Xavier Burgués[1], Xavier Franch [1]

[1]L.S.I.-U.P.C. (Departament Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya)
e-mail: diafebus|franch@lsi.upc.es

Resumen: Se propone una metodología para el diseño e implementación de estructuras de datos complejas, con el objetivo de paliar la ausencia de propuestas en este campo. La metodología está compuesta por cuatro etapas: análisis del enunciado, modelización del dominio, identificación de tipos abstractos de datos componentes y optimización de la estructura.

1.- MOTIVACIÓN.

El contenido habitual de las asignaturas de estructuras de datos de las licenciaturas e ingenierías en informática distingue claramente dos partes, como mínimo: el estudio de un repertorio versátil de tipos abstractos de datos (*TADs*) junto con su implementación individual mediante técnicas eficientes de representación, y una aproximación a la problemática del diseño e implementación de nuevas estructuras de datos (*EDs*), medianamente complejas, combinando dichos TADs invididuales para formar la solución.

Si bien para la primera parte existen un sinfín de textos de indudable calidad (especialmente por lo que a las EDs se refiere; no tanto a su estudio como implementación de TADs), no podemos decir lo mismo de la segunda. El diseño de nuevas EDs es un problema que algunos de estos libros o bien no abordan o bien tratan enumerando una serie de ideas generales que el programador debe ser capaz de aplicar a su contexto concreto, sin disponer de unas pautas bien definidas que lo guíen.

El objetivo de este artículo es proporcionar una metodología rigurosa para el diseño de nuevas EDs. Por "rigurosa" entendemos: una metodología con unas etapas bien definidas, usando unas notaciones con un significado claramente establecido. La metodología consta de cuatro etapas: análisis del enunciado, modelización del dominio, identificación de TADs componentes y optimización de la ED resultante.

2.- ANÁLISIS DEL ENUNCIADO.

El objetivo de esta primera etapa consiste en dejar explícitas todas aquellas cuestiones del enunciado relevantes al diseño de la ED. Sin considerar las cuestiones habituales en una fase de análisis de los requisitos (resolución de ambigüedades, detección de inconsistencias, etc.), nos interesa aquí:

- Determinar las operaciones aplicables sobre el nuevo TAD, es decir, su signatura. Estas operaciones deben tener un comportamiento claramente definido, ya sea mediante una especificación formal, ya sea mediante una explicación detallada de su funcionamiento.
- Detallar las características de los datos involucrados en el nuevo TAD. Básicamente, qué se sabe sobre su volumen (conocido o desconocido y, en el primer caso, una cifra aproximada o bien una calificación – "muchos", ...) y también cualquier otra característica que pueda influir en el diseño o implementación de la ED (por ejemplo, intervalo válido de valores de un campo numérico).
- Identificar los requisitos no funcionales de la ED. Básicamente, nos referimos aquí a la eficiencia esperada de las operaciones y a restricciones del espacio ocupado. La precisión puede variar de un problema a otro, desde requisitos muy precisos ("la operación de consulta debe ser de coste constante") hasta otros informales ("debe favorecerse el tiempo de ejecución de las operaciones de consulta").

Por ejemplo, consideremos el enunciado siguiente:

> Se quiere gestionar el uso de videoclips en una emisora de TV. Cada video tiene un título que lo identifica, una duración expresada en minutos y está editado por una compañía. Debe ser posible añadir elementos y listar los existentes. Asímismo, se debe dar soporte a la política de emisión siguiente para elegir un videoclip de una

duración dada: si hay videoclips no emitidos, se escoge el último adquirido; si no, se escoge el que hace más tiempo que se emitió.

Su análisis podría resultar en la información siguiente (por motivos de espacio, se abrevia la descripción de la signatura):

OPERACIONES:

crea()	
añade(v: video)	
cuál_toca(dur: nat): video	{ sigue la política de emisión de la emisora }
emite(dur: nat)	{ anota la emisión del videoclip que toca }
lista(comp: cadena): lista	{ lista de videoclips de una compañía (orden alfabético) con su número de emisiones }

REQUISITOS E INFORMACIÓN NO FUNCIONAL:

La duración máxima de un videoclip es de 4 horas
Añade, cuál_toca y *emite* deben ser eficientes
Hay un número desconocido y elevado de videoclips
Hay un número desconocido y pequeño de compañías

Como resultado de esta etapa, pues, se dispone de un enunciado completamente elaborado que sirve de punto de partida para el diseño e implementación de la ED.

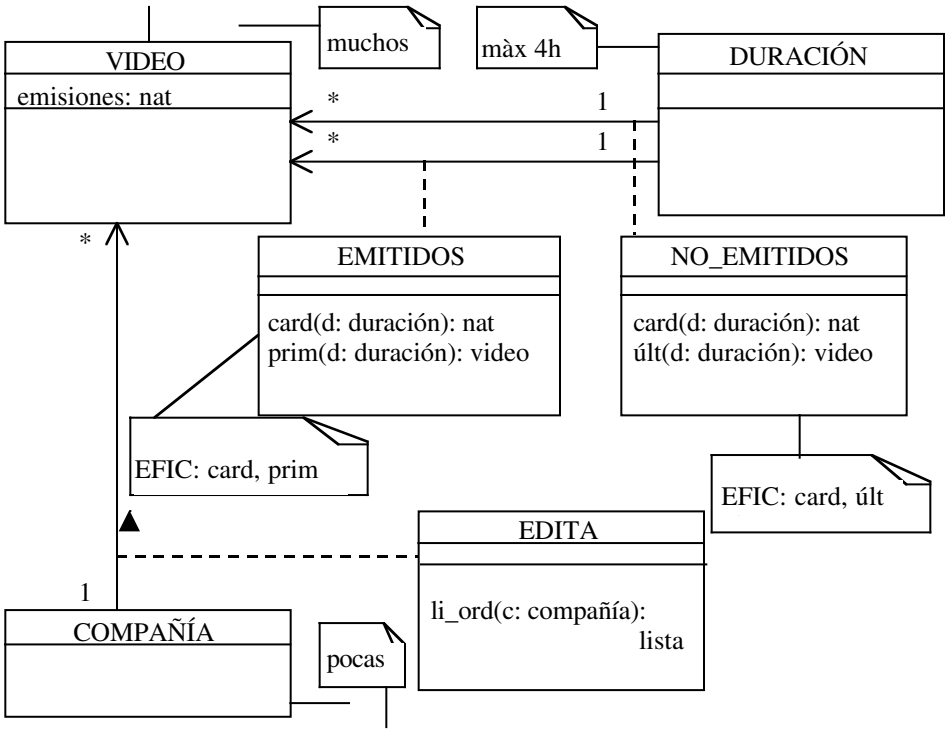
3.- MODELIZACIÓN DEL DOMINIO.

El siguiente paso consiste en identificar todas aquellas entidades que intervienen en la estructura y establecer sus interrelaciones. Para ello, usaremos como base una notación consolidada en el campo de la ingeniería del *software* (IS), en este artículo UML [BRJ99], aunque la metodología puede emplearse igualmente utilizando otra notación similar, por ejemplo *Entity-Relationship*. Esta opción es lógica, ya que el diseño e implementación de EDs no es más que un caso particular de IS y por ello podemos aprovechar los conocimientos en este campo. Como ventaja adicional, el uso de la misma notación en asignaturas diferentes minimiza el esfuerzo de aprendizaje del alumno y proporciona una mayor integración entre los campos de las EDs y de la IS.

El proceso de esta segunda etapa es, pues:

§ Construir un diagrama de clases UML que incluya todas las colecciones de datos del problema (representadas mediante pares de clases colección/elemento) y las diversas interrelaciones (asociaciones, en terminología UML) entre estos elementos. Debe destacarse aquí que algunas de estas clases y asociaciones pueden no ser relevantes para la construcción de la ED, pero aun así se recomienda incluirlas para una mayor compleción del modelo resultante.

§ Añadir en este diagrama todas las características identificadas en la segunda subetapa de la fase anterior y también las restricciones no funcionales referentes al espacio de la ED. La mayoría de esta información irá asociada a las clases mismas.



§ Escribir el pseudocódigo de las operaciones de la signatura del TAD, comenzando por las consultoras, pues son éstas las operaciones que determinan la información mínima que debe mantenerse en la estructura. Durante este proceso, aparecerán operaciones auxiliares que deberán asociarse a las clases y asociaciones del dominio, según

establece la notación usada. Debe incluirse la signatura de estas operaciones, así como una descripción de su comportamiento. Asímismo, deberán trasladarse los requisitos no funcionales de las operaciones del TAD sobre estas nuevas operaciones, para posibilitar en futuras etapas una implementación acorde con los requisitos establecidos. Estas operaciones auxiliares deberán ceñirse a un catálogo predefinido de operaciones, formado por todas aquellas propias de los TAD vistos en la asignatura. Nos referimos, pues, a las diversas modalidades de operaciones de inserción, supresión, consulta y listados, principalmente.

La figura de la página anterior muestra parte del diagrama de clases obtenido después de considerar las operaciones *cuál_toca* y *lista* del ejemplo.

4.- IDENTIFICACIÓN DE LOS TADS COMPONENTES.

Una vez se dispone de una descripción precisa y completa de las diferentes partes que componen la estructura, es el momento de elegir TADs conocidos para implementarlas, junto con sus estrategias adecuadas de representación. Dispondremos de los TAD habituales para implementar tanto las clases como las asociaciones [FRA97], si bien el TAD *Relación* y los grafos sólo apareceran en asociaciones, por involucrar dos dominios de datos. En cualquier caso, existen básicamente cuatro posibilidades:

- Reutilizar el TAD íntegramente, sin más cambios que un posible renombramiento de sus operaciones. Por desgracia, no acostumbra a ser el caso usual.
- Extender el TAD para añadirle nuevas funcionalidades. Si la asignatura se presenta con el paradigma de la orientación a objetos, puede usarse el mecanismo de herencia a tal efecto.
- Si las funcionalidades son muy diversas, puede ser posible identificar más de un TAD como componente de la clase/asociación.
- Excepcionalmente, deberá crearse un TAD *ad hoc* para la clase.

En nuestro ejemplo, el modelo presentado nos lleva a la elección de pilas y colas para los videoclips emitidos y no emitidos, respectivamente. Asimismo, dados los requisitos sobre ellos, los videos deben guardarse en un conjunto implementado con un AVL, las duraciones en una tabla

implementada mediante un vector y las compañías en una lista. Asímismo, la asociación *edita* se resolverá usando listas.

Como resultado de este paso, se dispone de un entramado de TADs que responde tanto a la funcionalidad como a la no funcionalidad requeridas. Puede decirse que el diseño ha terminado, pero no así la implementación.

5.- OPTIMIZACIÓN DE LA ESTRUCTURA.

Para completar la implementación del TAD, debemos detectar y solucionar posibles ineficiencias y tomar las decisiones necesarias para que todo detalle de representación quede resuelto. El resultado de este paso es pues la finalización del diseño de la ED, quedando tan solo pendiente la codificación de la misma. Principalmente:

- Cualquier redundancia de espacio provocada por la repetición de información puede resolverse con la fusión de las EDs involucradas.
- Cualquier acceso directo a una estructura puede hacerse lo más eficiente posible mediante apuntadores directos a las posiciones que ocupan los elementos.
- Deben completarse los detalles necesarios para la completa representación del tipo y las operaciones, por ejemplo: organizaciones usadas en las tablas de dispersión; funciones de dispersión utilizadas; elección de memoria dinámica o vectores; etc.

Notemos que los dos primeros puntos pueden provocar pérdida de modularida, pues se accede directamente a la implementación de los TADs combinado, obteniendo pues código menos robusto. Por lo tanto, debe evaluarse con detenimiento si la optimización de la estructura realmente justifica esta pérdida de claridad (en [MF97] puede encontrarse una metodología que intenta solucionar este problema).

En nuestro ejemplo, dado el desconocimiento del número de ítems a guardar, decidimos usar memoria dinámica para las diversas EDs que aparecen. Por otro lado, parece recomendable sustituir las apariciones de videos en pilas/colas/listas por punteros a su posición en el AVL, para ahorrar espacio.

6.- REFERENCIAS.

[BRJ99] Booch, G.; Rumbaugh, J.; Jacobson, I. *The Unified Modeling Language User Guide*. Addison-Wesley 1999.

[FRA94] Franch, X. *Estructuras de Datos: Especificación, Diseño e Implementación*. Edicions UPC, colección Politext 30, 1994. Accesible electrónicamente en http://www.edicionsupc.es.

[MF97] Marco, J.; Franch, X. *Reconciliando modularidad y eficiencia mediante atajos*. Actas de las III Jornadas de Informática de la A.E.I.A., El Puerto de Sta. María (Cádiz), julio 1994.