Informática Educativa Vol. 6 Nº 1, 1993
Proyecto SIIE, Colombia pp. 45-54

CONSTRUCCION DE MECs CON DISEÑO ORIENTADO POR OBJETOS

María Elizabeth Mónica Carlier T.

RESUMEN

En este artículo se discute sobre el diseño computacional de Materiales Educativos Computarizados, MECs, utilizando la Metodología de Diseño Orientada por Objetos, MDOO. Se presentan los conceptos básicos de dicha metodología y se analiza su uso para el desarrollo MECs.

INTRODUCCION

El diseño de MECs debe garantizar su calidad, tanto desde la perspectiva educativa como informática [i]. La primera se logra realizando un diseño educativo comprometido con el objetivo que se quiere lograr, aplicando los principios psicológicos y comunicacionales más adecuados al tipo de objetivo y a la población a quien va dirigido el material. La clave para lograr calidad desde la perspectiva informática está en usar buenas técnicas de especificación y una metodología de diseño y programación que permita desarrollar en forma natural los factores de calidad de software [ii].

La Metodología de Diseño Orientado por Objetos, MDOO, se presenta como una alternativa para diseñar y desarrollar software de calidad. Es decir, software que sea [2, iii, iv]:

- eficiente: hace buen uso de los recursos
- confiable: hace bien lo que dice que hace
- amigable: fácil de usar
- modular: diseñado de forma que sea posible programarlo como ensamblaje de pequeñas piezas
- estructurado: el problema se subdivide de forma que se descubran los detalles de una manera progresiva (o jerárquica)
- reutilizable: posibilidad de utilizar los productos del software, completamente o en parte, en nuevas aplicaciones
- con parsimonia: no reinventa soluciones a problemas resueltos, aprovecha las existentes
- robusto: habilidad para funcionar en condiciones fuera de lo normal
- fácil de mantener: fase que viene después del desarrollo, sea en atención a cambios en la especificación o en los requerimientos.

Un MEC, Material Educativo Computarizado, se puede considerar como un software interactivo -programa que ofrece servicios y facilidades en pos de un objetivo según demandas de los usuarios. El diseño educativo de un MEC puede tener un enfoque algorítmico -guía la forma de actuar de docentes y estudiantes- o heurístico -favorece la invención y el descubrimiento- [1]. En el segundo enfoque necesariamente los usuarios -estudiantes y profesores- interactúan con el MEC como si lo estuvieran haciendo con objetos del mundo real; en el primero, siempre que el argumento se desarrolle alrededor de micromundos. Para lograr esta interactividad es necesario reflejar en el MEC la estructura del mundo al que se hace referencia (real o imaginario), lo que generalmente se consigue mediante el uso de micromundos. Se puede afirmar que en la mayoría de los MECs el uso de micromundos es relevante.

En software donde el manejo de micromundos es importante, el código correspondiente a la interfaz suele ser mayor que el resto del código de la aplicación. Por esto se debe tener mucho cuidado, en el diseño computacional, para separar los módulos correspondiente a la interfaz del resto de modulos, con el fin de favorecer factores de calidad de software como modularidad y mantenimiento, entre otros.

El problema consiste en que desarrollar software de calidad y hacerle sentir al usuario que está interactuando con un micromundo que representa una copia del mundo al que se hace referencia. No es tarea fácil. Es necesario contar con una buena metodología de diseño e implantación que permita que, tanto diseñadores como programadores, sigan una disciplina que les facilite lograr los factores de calidad del software en sus aplicaciones.

En este artículo se discute sobre el uso de la metodología de diseño orientada por objetos para el diseño computacional de MECs. En la primera parte se describen algunos conceptos básicos de la metodología y en la segunda se hace un análisis del uso de la misma para el desarrollo de MECs.

CONCEPTOS BASICOS DEL DISEÑO ORIENTADO POR OBJETOS

La MDOO conlleva identificar objetos del mundo en referencia con sus respectivas características y comportamientos, así como determinar protocolos de comunicación entre dichos objetos para que interactúen entre sí.

El objetivo central en el diseño es que todo elemento del mundo (objeto real) corresponda directamente a un objeto del mundo computacional (objeto formal) que lo modele [v].

Toda la teoría se encuentra basada en un conjunto pequeño de conceptos: Objeto, clase y método.

- Un *objeto* es la representación dentro del modelo computacional de un elemento del mundo real. Todo objeto tiene atributos y comportamiento.

  Por ejemplo, para desarrollar un simulador de vuelo es necesario modelar un avión, entre otros objetos.

Objeto: avión

Atributos: marca, modelo, color, piloto, copiloto, tablero de mandos, sillas, caja negra, puertas, etc.

Comportamiento: Arrancar, despegar, inclinarse, subir, bajar, aterrizar, etc.

- Una *clase* agrupa objetos que por sus características comunes pueden ser representados de una manera similar.

  Por ejemplo existen diferentes tipos de aviones: comerciales, de guerra, de carga, etc. Todos los aviones comerciales se caracterizan por los mismos atributos y tienen un comportamiento propio, por lo cual se podría tener una clase "aviones comerciales". Igual sucede con los aviones de guerra, por lo cual se podría tener una clase aviones de guerra y así sucesivamente.

- Un *método* modela el comportamiento de los objetos pertenecientes a una clase.

  Por ejemplo, dentro de la clase avión se puede tener los siguientes métodos: arrancar, subir, bajar, aterrizar, etc.

Utilizando estos conceptos se pueden realizar diseños que atiendan los diferentes factores de calidad de software.

Por ejemplo: el concepto de clase favorece el factor de modularidad, ya que cada clase que se defina podría constituir un módulo. También favorece el factor reutilización, porque cada clase se puede constituir en módulos independientes que en el futuro puedan usarse en otras aplicaciones. De igual forma estos conceptos van favoreciendo otros factores.

A continuación se explicarán en detalle los conceptos de modularidad y reutilización.

- Modularidad: Como ya se dijo, consiste en construir programas como ensamblaje de pequeñas piezas.

  Existen diferentes criterios de modularidad.

  - Se puede descomponer un problema en muchos subproblemas, cuya solución pueda ser encontrada separadamente.

    Ejemplo:

    Problema: vuelo de un avión.

    Subproblemas: el problema del arranque, el del despegue, el de subir, el de mantenerse a una velocidad constante a cierta altura, el de descender y el de aterrizar.

Solución: una vez resuelto cada uno de estos subproblemas se tiene la solución total del problema original: el vuelo de un avión.

- Se pueden tener módulos que pueden ser libremente combinados con otros, para producir nuevos sistemas. Se deben diseñar piezas que desarrollen trabajos bien definidos y sean utilizables en diferentes contextos. Esto favorece la reutilización.

  Ejemplo: Se va a desarrolar un MEC donde el micromundo está conformado por objetos voladores (no necesariamente aviones). Se pueden utilizar los métodos definidos en el simulador de vuelo para solucionar el problema de vuelo del avión, para hacer volar dichos objetos voladores.

Existen varios criterios de descomposición modular que pueden ser determinados por el diseñador del sistema, de acuerdo con sus objetivos y conveniencia.

• Reutilización: está asociada con los diferentes criterios de modularidad que se utilicen. Es decir, si se definen módulos independientes con labores bien definidas, es posible usarlos en el desarrollo de otras aplicaciones.

Aquí también se debe mencionar el tema de la *herencia*. La herencia es un mecanismo por el cual se pueden aprovechar el diseño y la implantación de clases ya existentes para definir nuevas clases.

Ejemplo: Supóngase que se tiene la clase *avión* ya especificada -atributos y comportamiento- y se necesita especificar las clases: avión comercial, avión de guerra y avión de carga.

Como todos los aviones, independientemente del tipo (comercial, guerra, carga) tienen algunos atributos y comportamientos similares (atributos: marca, modelo, color, piloto, copiloto; comportamiento: despegar, volar y aterrizar), no es necesario volver a definir cada uno de los atributos y comportamientos de la clase avión para cada uno de los diferentes tipos de aviones. Mediante el mecanismo de herencia, las clases: avión comercial, avión de guerra y avión de carga pueden heredar los atributos y comportamientos de la clase avión. Para calificar cada clase específica se le definen los atributos y comportamiento particulares a cada una de ellas.

Esto se puede representar en la siguiente gráfica:

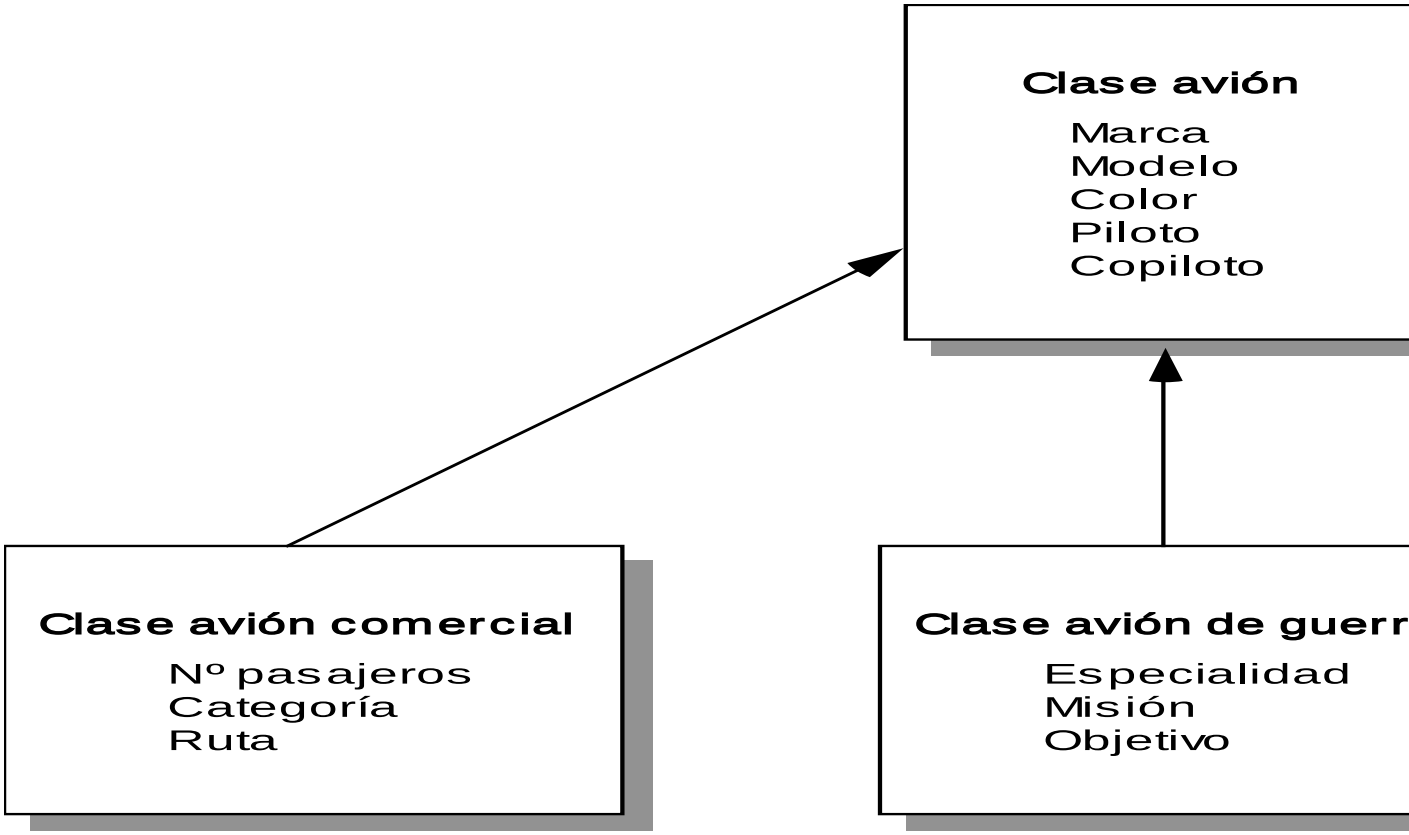

Figura 1. Ejemplos de herencia

La gráfica expresa que la clases avión comercial, avión de guerra y avión de carga heredan todos los atributos y comportamiento de la clase avión.

METODOLOGIA DE DISEÑO ORIENTADO POR OBJETOS

Esta metodología se plantea en [4]. El proceso de diseño se puede resumir de la siguiente forma: análisis del mundo real, determinar las condiciones de validez del mundo; identificar y especificar los métodos de cada clase.

Análisis del mundo real

Se debe entender con claridad el mundo real del problema y describir el problema en términos de su mundo. Aquí están los objetos que existen en el mundo en el cual ocurre el problema, los objetos reales. El resultado de este análisis es una jerarquía de niveles de abstracción de los objetos que hacen parte del mundo. Analizando dicha jerarquía se puede empezar a determinar la herencia. En este momento se tiene un conjunto de clases con sus atributos.

Determinar las condiciones de validez del mundo

Consiste en determinar bajo qué condiciones el modelo del mundo es válido (correcto).

Identificar Métodos

Consiste en modelar el comportamiento de cada clase. Se especifican para cada clase sus respectivas operaciones.

Ejemplo: diseñar parte de un simulador de vuelo.

Dentro del mundo real de los aviones, existen aviones, pistas, espacio para volar, aeropuertos, torres de control, etc. En este ejemplo sólo se va a presentar un resumen del modelaje de la clase avión.

ANÁLISIS DEL MUNDO REAL

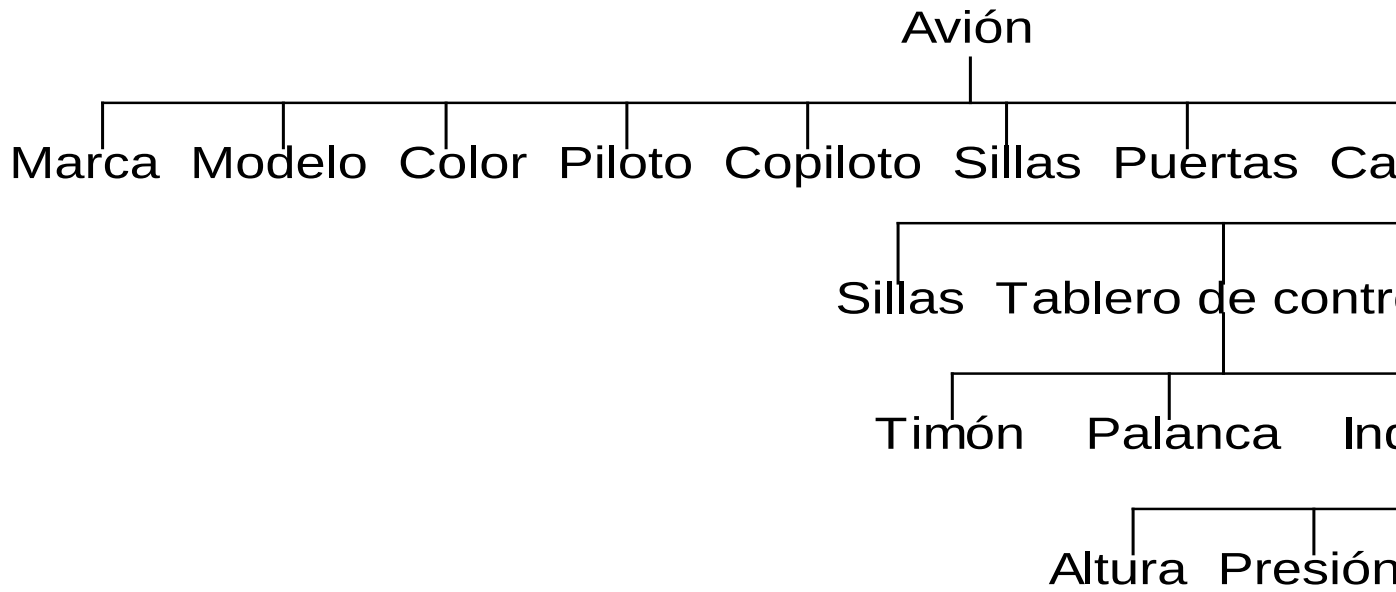

Figura 2. Jerarquía que subyace al objeto avión

Esta jerarquía representa todos los objetos que hacen parte del objeto avión. La semántica de dicha jerarquía es del tipo *parte de*. Es decir, se puede interpretar como: un avión tiene una marca, un modelo, un color, un piloto, sillas, cabina,.... La cabina tiene sillas, tablero de control, pedales, frenos. El tablero de control tiene timón, palanca, indicadores. Los indicadores tienen indicadores de altura, de presión, de oxígeno, de inclinación. Y asi se continuaría desarrollando la respectiva jerarquía.

DETERMINAR LAS CONDICIONES DE VALIDEZ DEL MUNDO

- Para volar se necesita un espacio libre.
- El avión necesita una pista despejada para arrancar.
- Un avión sin piloto no puede volar.
- Es necesario que tenga gasolina en el tanque de gasolina.
- Otras.

En resumen, aquí se definen las condiciones bajo las cuales el avión que se está modelando va a volar. Estas condiciones dependen del comportamiento natural de los objetos en el mundo real y de los objetivos que se quieran lograr con el MEC.

IDENTIFICAR Y ESPECIFICAR LOS MÉTODOS DE CADA CLASE

Se definen las operaciones sobre todos los objetos del mundo. Es decir, sobre todos los objetos que se modelaron en el primer punto.

Ejemplo.

Objeto: Avión

Atributos
- Marca
- Modelo
- .....

Métodos
- Crear avión
- Prender avión
- Despegar avión
- Subir avión
- Bajar avión
- Aterrizar avión
- Otros.

Los métodos salen del análisis del comportamiento de los objetos en el mundo real. Los nombres de los métodos los da el diseñador del sistema.

Finalmente, el diseñador debe llegar a comprometerse con la implantación de su modelo en el computador. Esto significa que tiene que definir tipos de datos para cada objeto, estructuras de datos, manejo de memoria principal y secundaria, etc.

LA MDOO PARA EL DESARROLLO DE MECs

No basta con aplicar la MDOO para desarrollar un MEC de calidad. En un material educativo la calidad está conformada por la calidad informática y la calidad educativa. Por consiguiente, es necesario enriquecer la metodologia planteada en la sección anterior con el diseño educativo. Es decir, una vez se tenga especificado el modelo del mundo real -atributos y comportamiento- es necesario incluir dentro de dicho modelo todos los detalles del diseño educativo.

Por ejemplo, si el objetivo del simulador de vuelo es que el usuario -el alumno o el profesor- descubra heurísticas, principios, para no estrellarse, se deben modelar otro tipo de objetos, por ejemplo, obstáculos. Y dentro de estos se pueden modelar objetos que no existan en el mundo real, v.gr., platillos voladores, que el avión tiene que esquivar para no estrellarse. De igual forma sucede con los métodos. Estos no solamente deben describir el comportamiento de los objetos del mundo real, también deben describir los objetivos educativos.

Con la salvedad anterior, se puede afirmar que la MDOO es ventajosa para el desarrollo de MECs por lo siguiente:

- Teniendo en cuenta que para determinar si se logra suplir la necesidad para la cual fue desarrollado todo software educativo debe ser sometido a prueba, con estudiantes y profesores, se puede afirmar que, una vez se concluya la primera versión del material, éste entra a una etapa de mantenimiento.

  Es necesario ajustar el software de acuerdo con los resultados de las pruebas. La discusión de calidad de software no se puede centrar en la fase de desarrollo. Se estima que un 70% del costo del software pertenece a mantenimiento. En el mantenimiento hay labores nobles e innobles. Nobles: cambios en la especificación y los requerimientos. Innobles: corrección de errores [vi].

  De esta forma, es evidente la necesidad de realizar un diseño e implantación de las aplicaciones educativas que esté comprometido con los diferentes factores de calidad de software, para que los cambios necesarios sean en la mayoría nobles.

- En el desarrollo de MECs de tipo heurístico el aprendizaje se produce por discernimiento repentino a partir de conjeturas acerca de situaciones experienciales, por descubrimiento. De este modo, es necesario reflejar la estructura del mundo real en el modelo computacional, para hacerle sentir al usuario, alumnos y profesores, que está viviendo una experiencia casi real y pueda llegar a la adquisición del conocimiento, con la orientación e interpelación por parte del profesor, por supuesto.

  La MDOO permite crear un modelo computacional que sea una imagen de la estructura del mundo real y, por consiguiente, permite reflejar esto naturalmente en el sistema.

- En el desarrollo de la mayoría de MECs el micromundo representa un porcentaje significativo con respecto al resto del código de la aplicación. Esto sugiere que se debe usar una metodología de diseño que permita separar la interfaz del resto del código con el fin de favorecer el mantenimiento del mismo.

- La reutilización es una característica que parece ser muy interesante dentro del desarrollo de software educativo. Hace pensar que, en un futuro, los profesores mismos podrán desarrollar su propio software. Los informáticos se encargarían de proporcionar la mayoría de las piezas de software para que los profesores puedan ensamblar sus propios programas, de acuerdo con sus propias necesidades.

CONCLUSIONES

La metodología de diseño orientado por objetos se presenta como una alternativa para desarrollar MECs de calidad y para ofrecer al usuario -alumnos y profesores- un micromundo para interacción que permita reflejar en forma natural el mundo real. NO es la única alternativa para desarrollar MECs con calidad.

Aplicando la MDOO y tomando en cuenta los factores de calidad, se logra desarrollar MECs con calidad en términos informáticos, pero no se garantiza la calidad educativa. Esta la garantiza el diseño educativo que se haga del mismo. Por supuesto se deben integrar tanto el diseño informático como el diseño educativo para lograr obtener un buen MEC. Por consiguiente, es necesario enriquecer la metodología de diseño orientado por objetos con el diseño educativo, para lograr desarrollar MECs de calidad.

Si bien es cierto que con cualquier metodología de desarrollo de software se puede elaborar software de calidad, también es cierto que no todas las metodologías son apropiadas en todos los casos. La MDOO permite hacer un modelaje más natural del mundo del problema en aplicaciones gráficas interactivas, en aplicaciones donde es necesario guardar una relación uno a uno entre el modelo real y el modelo formal, en aplicaciones dinámicas -que evolucionan rápidamente a través del tiempo.

Un MEC de tipo heurístico se puede considerar como una aplicación gráfica interactiva; en él es necesario reflejar la estructura del mundo real en el micromundo, lo cual permite afirmar que la MDOO es adecuada para desarrollar este tipo de software educativo. Para el desarrollo de MECs algorítmicos se podría aplicar cualquier metodología de diseño. Si un MEC de este tipo requiere manejo de micromundos, se recomienda utilizar la MDOO.

Utilizando la MDOO en el desarrollo de MECs y tomando como criterio de descomposición modular, módulos totalmente independientes y que realicen trabajos bien definidos, se podría ir pensando en el futuro en una fábrica de software educativo. En ella se proporcionarían todas las piezas de software necesarias para que los profesores desarrollen sus propias aplicaciones, de acuerdo con sus propias necesidades.

REFERENCIAS

[i] GALVIS, AH (1992). *Ingeniería de Software Educativo*. Bogotá: Ediciones Uniandes.

[ii] MEYER, B. (1988) *Object-Oriented Software Construction.* Prentice Hall.

[iii] PRESSMAN, R. (1988). *Ingeniería del Software. Un enfoque Práctico*. Mc. Graw Hill.

[iv] MUÑOZ, C., BARROS, R. (1991) *Implantación de Diseños Orientados por Objetos.* Bogotá: Uniandes, CIFI (memo de investigación Nº 66, mimeografiado).

[v] CHICAIZA, L., VILLALOBOS, J. (1991) *Hacia una metodología orientada por objetos para el diseño de aplicaciones gráficas interactivas*. Bogotá: Uniandes, CIFI (memo de investigación Nº 64, mimeografiado).

[vi] MEYER, B. (1989) *From Structured Programming to Object-Oriented Desing: The Road to Eiffel.* Structured Programming.