Una Metodología Orientada por Objetos para el Diseño de Aplicaciones Gráficas Interactivas

Luis B. Chicaíza S.
Jorge A. Villalobos S.

Grupo de Investigación DFAC
Departamento de Ingeniería de Sistemas y Computación
Universidad de los Andes
e-mail: lchicaiz@andescol.bitnet
jvillalo@andescol.bitnet
Apartado Aéreo 4976 - Bogotá, Colombia

Resumen: Se propone un modelo conceptual y una metodología de diseño para aplicaciones gráficas interactivas. El propósito es mantener separados la interfaz y el control del resto de la aplicación, con el fin de facilitar su mantenimiento. Se propone el enfoque orientado por objetos como una manera natural de diseñar y especificar cada uno de los elementos del software gráfico. Las aplicaciones gráficas se caracterizan porque el usuario puede interactuar con las imágenes de los objetos del mundo en el cual ocurre el problema y las manipulaciones que él haga sobre cualquiera de sus imágenes deben generar manipulaciones sobre los objetos representados.

Descriptores: Programación Orientada por Objetos, Ingeniería de Software, Aplicaciones Gráficas Interactivas, Interfaces-Usuario

1. Introducción

La programación orientada por objetos (POO) se ha convertido en una de las herramientas que más expectativas genera en el ámbito de la ingeniería de software, debido, posiblemente, al potencial que encierra para aumentar la calidad de las aplicaciones y a su notable capacidad para estructurar el software con miras a facilitar su posterior mantenimiento, uno de los más grandes escollos en las metodologías tradicionales de desarrollo. En las aplicaciones gráficas interactivas se requieren extensiones a la metodología, en lo que respecta a la arquitectura del software obtenido, puesto que los elementos gráficos y de interacción no se acoplan de manera natural con el resto de los componentes de un programa.

Una *aplicación gráfica interactiva* (AGI) maneja una representación gráfica del estado del mundo del problema, siempre actualizada, y el usuario debe poder interactuar con esa imagen como si lo estuviera haciendo con los objetos del mundo que ésta representa. Además, puede haber más de una imagen asociada con un objeto del mundo, y se crean problemas de validez y coherencia entre ellas. Cada una de estas posibles imagenes se denomina una *vista*.

El objetivo de este artículo es mostrar un modelo conceptual para las AGI, con el fin de situar estructuralmente cada una de sus partes y definir sus relaciones, para proponer una metodología que permita abordar de manera sistemática el problema de diseñar este tipo de aplicaciones.

2. Arquitectura del Modelo

La idea fundamental de la POO es copiar la estructura y el comportamiento del mundo en el cual ocurre el problema, con el fin de obtener un modelo operacional de dicho mundo. Una vez obtenido este modelo, es posible estimularlo para obtener una reacción que se pueda interpretar como la solución del problema planteado. Bajo este esquema, la interfaz de una aplicación interactiva se puede ver como un administrador de ese modelo, que mantiene el diálogo con el usuario, y que va interpretando sus comandos para traducirlos en el conjunto de estímulos necesario para que el modelo responda y se llegue a un nuevo estado de la aplicación. Esto se ajusta bastante bien a las propuestas de separación entre la interfaz y el resto de la aplicación que, con el propósito de facilitar el mantenimiento, sugieren, entre otros, [OLS83], [SCH85] y [VIL87].

Una AGI está compuesta por un modelo operacional del mundo en el que ocurre el problema y uno o varios administradores de dicho modelo, que permiten a los diferentes tipos de usuario manejar vistas de los elementos del mundo. Cada administrador controla el diálogo con un usuario y mantiene una representación gráfica coherente con el estado del modelo.

2.1 Modelo detallado

Si se consideran las funciones del administrador, se puede llegar a un modelo más detallado, en el que se separen el control, la interacción y el manejo de la representación gráfica del mundo. A grandes rasgos, el modelo opera así: los eventos son tomados por los elementos de la interacción e interpretados; se envía dicha información al control de diálogo y éste se encarga de propagar sobre los cuatro mundos (incluido él) las consecuencias de esta orden del usuario. Esto quiere decir que el control es el encargado de mantener la coherencia y de controlar toda la aplicación.

La interacción con el usuario se va a reflejar en una extensión al modelo, para permitir la estimulación externa de los objetos mediante eventos. La separación de mundos y la caracterización de los objetos de cada mundo van a facilitar el planteamiento de una metodología de diseño.

2.2. El Modelo del Mundo del Problema

En este modelo están los objetos que existen en el mundo en el cual ocurre el problema: *los objetos reales*. No tienen imagen ni es posible interactuar directamente con ellos. Son una copia en estructura y comportamiento de los elementos del mundo real que intervienen en el problema.

2.3. El Modelo de la Interacción

La misión de este mundo es manejar la interacción con el usuario a través del mecanismo de eventos, generados a partir de un conjunto de dispositivos. En este mundo viven elementos tales como menús, ventanas, barras de scroll, botones, diales, etc. Se puede modelar como un conjunto de objetos que se encuentran a la espera de una interacción de parte del usuario. Cada uno de estos objetos se denomina un *objeto interactivo*. Dentro del mundo de la interacción existe también un manejador de eventos, que es capaz de generar un mensaje al objeto con el cual el usuario quiere interactuar. Este manejador de eventos es la única extensión al modelo de objetos.

Desde un punto de vista funcional, el manejador de eventos recibe las entradas generadas desde los dipositivos, identifica el objeto interactivo que debe activar y le envía a éste un mensaje con toda la información necesaria para que actúe. Un objeto interactivo es un elemento abstracto, sin imagen (vive en un mundo que no visualiza directamente el usuario), creado casi siempre desde el mundo del control para permitir la interacción con el usuario.

Un objeto interactivo sabe siempre responder al mensaje que envía el manejador de eventos para activarlo, con un método que decide cuál acción tomar, utilizando para eso su repertorio de eventos. Este método se denomina su *despachador*. El objeto puede realizar cuatro tipos de acciones al recibir el mensaje que lo activa: ignorar el evento, consumir la entrada (reaccionar modificando sus atributos), interpretar la entrada y enviar un mensaje a su padre con esa interpretación, o enviar

directamente la información del manejador de eventos a su padre. Los objetos interactivos también tienen métodos de creación (reciben a su padre como parámetro) y métodos para manejar su estado.

2.4. El Modelo de la Visualización

En el mundo gráfico viven las imágenes de los objetos reales y de los objetos interactivos. Aquí se encuentran desde la representación gráfica de un menú hasta las diferentes vistas de un objeto real. Cada una de estas imágenes se denomina un *objeto gráfico* y son el medio para que el usuario pueda visualizar el estado de los diferentes modelos que componen una AGI..

A diferencia de los objetos reales y de los objetos interactivos, los objetos gráficos no son independientes: siempre están relacionados con un objeto de otro mundo que se denomina su *objeto mellizo*. En el caso más general, tiene un mellizo en el mundo de la interacción y otro mellizo en el mundo real, dando lugar a un objeto real con una imagen con la cual es posible interactuar. Que un objeto gráfico no sea independiente significa que sus atributos gráficos son calculables a partir de los atributos de su objeto mellizo. Esto implica que debe existir una función que, a partir de un subconjunto de los atributos del objeto mellizo, es capaz de calcular los atributos gráficos. Esta función se llama la *función directa de relación.*

Con esta función es posible construir y manejar las imágenes de los objetos interactivos y una o varias vistas de los objetos reales, pero todavía no es suficiente para permitir la interacción directa con la imagen de un objeto real. Para esto, es necesario exigir que la función de relación sea biyectiva, de manera que sea posible propagar directamente la interacción del usuario, y se generen cambios de estado en el objeto mellizo correspondiente. Puesto que es biyectiva, existe su inversa, llamada la *función inversa de relación*, que permite calcular los atributos de los objetos mellizos a partir de los atributos gráficos.

2.5. El Modelo del Control

Este modelo representa la parte funcional de la aplicación; es responsable de coordinar el flujo interno de control y de garantizar la coherencia de los distintos modelos. Su comportamiento y estructura se deben diseñar a partir de los requerimientos del usuario y especificar mediante algún formalismo, para luego poder modelar este mundo en términos de objetos.

Un formalismo muy utilizado para especificar el mundo del control es el de los *diagramas de transición de estados* (DTE) [TOR87], [OLS83], en los cuales, utilizando un grafo, se muestra el flujo de control al interior de la aplicación, y se hace explícito el conjunto de servicios disponibles para el usuario en cada momento. Dado el caracter miltiventanas y multiusuario de las AGI, es necesario un conjunto de DTE para diseñar el control, con uno de ellos como inicial.

2.5.1. Objetos del Control

Al modelar el mundo descrito por un DTE a través de objetos, se obtiene un conjunto de elementos llamados *objetos de control* (cada uno representa un nivel), que van a manejar el flujo de la aplicación utilizando el mecanismo de mensajes. El objeto de control responde a los mensajes de sus hijos con un método que decide la forma de reaccionar, lo mismo que el objeto de control que debe crear antes de destruirse. Este método se llama el *despachador*. Existe, además, un método interno por cada procesador.

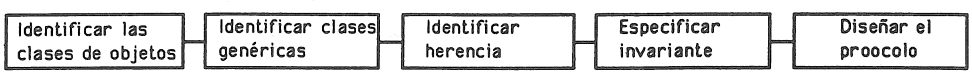
3. Una Metodología para Diseñar AGI

En esta parte se propone una metodología de diseño orientada por objetos que, partiendo de una fase previa de análisis, produce instancias válidas del modelo conceptual presentado anteriormente. Inicialmente se presenta una metodología general para diseño orientada por objetos, aplicable a cualquier mundo, y luego se harán especializaciones para ajustar la metodología a las características particulares de cada uno de los mundos de la aplicación.

3.1. Metodología de Diseño Orientada por Objetos

Una metodología no puede ser vista como un algoritmo infalible para obtener diseños óptimos, sino como una guía de apoyo para producir diseños de buena calidad. La metodología que se propone reconoce este hecho y ofrece un esquema ordenado para trabajar, dividido en una serie de etapas, y un conjunto de métodos asociado con cada una de ellas.

Se propone una metodología de cinco pasos:



3.1.1. Identificar las Clases de Objetos

En esta fase se identifican todas las clases de objetos necesarias para modelar el mundo del problema. Se parte del conocimiento obtenido en la etapa de análisis y se llega a una estructura del modelo, denominada el *grafo de niveles de composición* (GNC).

Los atributos de una clase son la información necesaria (objetos de otra clase) para abstraer y modelar el estado de un tipo de elemento del mundo. Se deben agregar atributos a una clase hasta que todas las características interesantes en el mundo del problema estén incluidas [MEY88]. Esta estructura da lugar a un grafo dirigido (el GNC del modelo) cuyos vértices son las clases y los arcos la relación de pertenencia (clase → atributo). Las clases de primer nivel son aquellas que no

tienen predecesores (clases externas) y las de último nivel las que no tienen sucesores (clases de base del sistema).

Se proponen dos formas de identificar las clases de objetos y crear el GNC. Una, utilizando el enfoque de arriba hacia abajo [JAL89], y la otra avanzando en el proceso de identificación de abajo hacia arriba [MEY88], aunque las combinaciones de las dos resultan muy adecuadas.

Método de arriba hacia abajo: 1) identificar los objetos de primer nivel tomando como guía aquellos que están directamente involucrados en el problema y que se pueden considerar como los más "grandes" en el mundo , 2) para cada una de las clases identificadas modelar su estado a través de un conjunto de atributos y adicionar las clases de estos al grafo de niveles de composición, 3) repetir el segundo paso hasta llegar a las clases de último nivel.

Método de abajo hacia arriba: 1) partiendo de las clases de último nivel construir el siguiente nivel de composición, e identificar las clases que se pueden modelar con atributos que ya están en el grafo, 2) repetir el paso uno hasta llegar a los objetos de primer nivel.

3.1.2. Identificar las Clases Genéricas

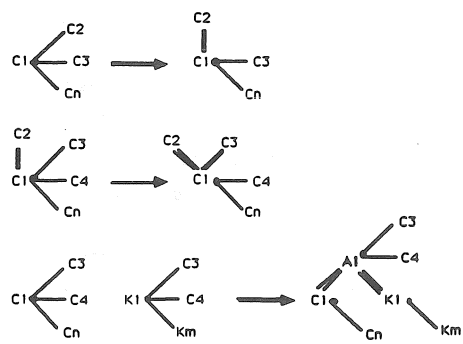
La entrada a esta etapa es el GNC del modelo y la salida es una nueva versión del GNC, con algunas de las clases parametrizadas, y un conjunto de esquemas de clase que las soportan.

3.1.3. Identificar las Relaciones de Herencia

La entrada a esta etapa es el GNC que se obtiene de la etapa anterior y la salida es el GNC aumentado con las relaciones de herencia (GNCH). Se coloca un arco de herencia entre dos clases del GNC (clase1 → clase2) si la clase2 hereda de la clase1.

Método de análisis sintáctico:

- buscar en el GNC alguno de los patrones del lado izquierdo de las reglas de reescritura:



- evaluar si en el mundo la relación de herencia que sugiere la regla tiene sentido, es decir, que sus comportamientos coinciden (análisis semántico)
- aplicar la regla de reescritura sobre el GNC para identificar la herencia simple (regla 1), la herencia múltiple (regla 2) y las clases abstractas (regla 3)

3.1.4. Especificar el Invariante de las Clases

El *invariante* de una clase es un predicado sobre el estado de un objeto que dice si éste es válido en el modelo. Se define así un subespacio sobre el espacio de todos los estados posibles de un objeto, en el que únicamente existen elementos que modelan objetos válidos en el mundo del problema. Se propone el siguiente método para construir el invariante de cada clase: 1) recorrer de arriba hacia abajo las jerarquías de herencia y escribir el invariante de cada clase, 2) para cada clase utilizar el GNC para identificar sus atributos y colocar una aserción que relacione los valores de los atributos y defina condiciones de validez de cada uno de ellos, 3) si la clase hereda de otras, construir el invariante a partir de la conjunción de los invariantes de las clases de las cuales hereda, y agregar una aserción que imponga nuevas restricciones y/o relaciones entre los atributos nuevos.

3.1.5. Diseñar el Protocolo de las Clases

Al exterior del modelo los únicos métodos visibles deben ser aquellos de las clases de primer nivel. No es posible utilizar los servicios de los atributos directamente porque, en tal caso, se podría perder la coherencia: sería posible alcanzar un estado en el cual el atributo cumpliera el invariante de su clase, pero el objeto que lo posee no cumpliera con el suyo. Esta etapa se divide en tres partes: identificar las operaciones de cada clase, verificar la validez y la completitud del diseño, especificar cada método. Para identificar las operaciones de cada clase se proponen tres métodos:

Método basado en la teoría de Tipos Abstractos de Datos: Ver [CAR91].

Método de Propagación de Necesidades: 1) identificar los servicios que debe proveer el modelo hacia el exterior y asociarlos con operaciones de las clases de primer nivel, 2) propagar dentro del GNC las necesidades que se van generando de una clase hacia sus atributos

Método de Propagación de Servicios: 1) especificar las operaciones de los objetos de último nivel, pensando en los servicios que este tipo de elementos debe prestar en el mundo, 2) propagar hacia arriba los servicios que ofrecen las clase de bajo nivel, teniendo en cuenta los servicios que ofrecen los atributos para diseñar las operaciones de cada clase

Para verificar la validez y completitud del diseño se debe tener en cuenta: 1) cada método de cada clase debe ser expresable en términos de los métodos de sus atributos, 2) los métodos de las clases

del ultimo nivel deben ofrecer todos los servicios externos que se requieren desde otros modelos. Para especificar un método se construye su precondición y su postcondición.

3.2. Diseño Orientado por Objetos de AGIs

En esta parte se especializa la metodología planteada en la sección anterior, para ajustarla al diseño de cada uno de los mundos que conforman una AGI.

3.2.1. Diseño del Modelo del Mundo

El método que se propone corresponde exactamente a la metodología general presentada; la única precisión que se debe hacer es en el diseño del protocolo, en el cual, los servicios externos al modelo se deben definir a partir de los requerimientos establecidos en el modelo de control, haciendo necesario avanzar en este segundo modelo para poder evaluar completitud

3.2.2. Diseño de la Interfaz

Se utiliza la metodología general propuesta, con las siguientes precisiones:

- utilizar los recursos (librerías, etc.) que ofrece el sistema como clases de último nivel al crear el GNC del modelo
- sólo clases interactivas y tipos simples pueden pertenecer al GNC del mundo de la interfaz
- para cada objeto interactivo: 1) determinar y codificar los eventos a los cuales debe responder, 2) la forma de reaccionar y 3) asociar un método con cada evento, teniendo en cuenta que para los objetos interactivos compuestos deben incluirse los eventos de sus componentes
- las constructoras deben recibir como argumento el objeto que lo crea.
- El protocolo consta de métodos contructores, de reacción a eventos (uno por cada evento), y el despachador.
- La especificación de un método de reacción a eventos, incluye en la postcondición el tipo de acción que toma como respuesta al evento que lo activa. La manera de notificar a su padre es llamar su método despachador.

3.2.3. Diseño del Control

La entrada a esta etapa es el conjunto de categorías de usuarios, el conjunto de requerimientos y el bosquejo de la interacción. Se propone una metodología de dos etapas:

- construir el conjunto de DTE
- traducir los DTE a objetos

3.2.4. Diseño de la Representación Gráfica

El propósito de esta etapa es determinar la manera como se van a visualizar los objetos de la AGI. Las entradas son el bosquejo de la interacción que desea el usuario, la información sobre las vistas,

el conjunto de DTE y los modelos del mundo real y de interacción. Se propone un método que procesa todos los elementos del conjunto de DTE en los siguientes pasos:

- determinar el conjunto de vistas
- diseñar los objetos gráficos
- establecer las funciones de relación
- enriquecer los modelos real e interactivo.

4. Conclusiones

En este artículo se presenta una propuesta de metodología de diseño orientada por objetos para aplicaciones gráficas interactivas. Más que una solución al problema que plantea el desarrollo de este tipo de aplicaciones, se pretende establecer una metodología, con la cual sea posible enfrentar este problema de una manera sistemática. La metodología que se propone en el artículo ha sido probada en el desarrollo de software gráfico para programación de robots fuera de línea, que incluye todas las características de una AGI. En este caso se utilizó C++ como lenguaje de implantación.

6. Bibliografía

[CAR91] Cardoso, R., Verificación y Desarrollo de Programas, Ediciones Uniandes, agosto 1991.

[JAL89] Jalote, P., Functional Refinement and Nested Objects for Object-Oriented Design, IEEE Transactions on Software Engineering, vol. 15, No. 3, Marzo/89.

[MEY88] Meyer, B., Object-Oriented Software Construction, Prentice-Hall, 1988.

[OLS83] Olsen, D., Dempsey, E., SYNGRAPH: A Graphical User Interface Generator, Computer Graphics, Vol. 17, No. 3, 1983.

[SCH85] Schulert, A., Rogers, G., ADM - A Dialog Manager, CHI'85 Proceedings, 1985.

[TOR87] Toro, V., Diseño, Especificación y Prototipificación de Sistemas Interactivos, XIII Conferencia Latinoamericana de Informática, 1987.

[VIL87] Villalobos, J., Arquitectura de una Interfaz-Usuario: Un Modelo General, XIII Conferencia Latinoamericana de Informática, 1987.

[BUX83] Buxton, W., Lamb, M., Towards a Comprehensive User Interface Management System, Computer Graphics, Vol. 17, No. 3, 1983.