PATRONES DE DISEÑO PARA EL MODELADO DE SOFTWARE EDUCATIVO

Ricardo Armando Barrera Cámara*
José Enrique Álvarez Estrada

Resumen

Este trabajo describe una serie de patrones de diseño, descubiertos en el transcurso de una investigación, que facilitan el desarrollo de Software Educativo (SE) tomando como base el paradigma orientado a objetos: Patrón de Software Educativo Tutorial, Patrón Ejercitación y Práctica, Patrón Simulador y Juego Educativo, en donde se describe la estructura y comportamiento que caracteriza a tales software.

Introducción

Software educativo, programas educativos, programas didácticos, modelos educativos computarizados, se consideran sinónimos para designar genéricamente los programas para computadora creados con la finalidad específica de ser utilizados como medio didáctico, es decir, para facilitar los procesos de enseñanza y de aprendizaje.

Los SE, a pesar de tener unos rasgos esenciales básicos y una estructura general común, se presentan con unas características muy diversas: unos aparentan ser un laboratorio o una biblioteca, otros se limitan a ofrecer una función instrumental del tipo máquina de escribir o calculadora, otros se presentan como un juego o como un libro, bastantes tienen vocación de examen, unos pocos se creen expertos y, por si no fuera bastante, la mayoría participa en mayor o menor medida de algunas de estas peculiaridades. Se han elaborado múltiples topologías y/o jerarquías que clasifican a los programas educativos a partir de diversos criterios. Algunas concepciones son:

Tutorial: guían al alumno a través de una estructura secuencial y/o consecutiva en el aprendizaje de conocimiento, pudiendo estar en función a la edad, permitiendo así retroalimentación a través de un número determinado de ejercicios (evaluación) y resultados; son considerados clases especiales de software de ejercitación y práctica.

Ejercitación y práctica: aplican al estudiante varios ejercicios y lo retroinforman acerca del los resultados obtenidos, en los cuales se aplican algunas características como tiempo, despliegues en pantalla, castigos (quitar puntos), etc.

Simuladores y juegos educativos: apoyan al aprendizaje de forma experiencial, a través del aprendizaje por descubrimiento, parecido a una situación real, resolviendo problemas, aprendiendo procedimientos, llega a entender las características de los fenómenos y como controlarlos, pudiendo ser motivante y recibiendo información de retorno. El alumno trabaja por ensayo y error, probando cosas a ver qué resulta. Los juegos educativos presentan las mismas características de los simuladores, con la diferencia que pueden o no simular la realidad, se caracterizan por promover situaciones excitantes (retos) o entretenidas(entretenimiento).

Sistemas expertos con fines educativos: sistemas de computación capaces de representar. razonar, resolver problemas y dar consejos a quienes no son expertos, los cuales requieren experiencia humana, y trabajan sobre la base de reglas de inferencia obtenidas a partir de bases de datos y de las decisiones del usuario, pudiendo reconstruir y analizar el conocimiento del alumno y reorientarlo.

Tutoriales inteligentes: son una combinación entre los tutoriales y los sistemas expertos, deben incluir un modelo del estudiante, en el cual se plasman tanto los conocimientos, habilidades y destrezas que el alumno posee o demuestra, como un módulo tutor el cual crea las situaciones por resolver, evaluar o diagnosticar al alumno.

El Modelado Orientado a Objetos nos permite describir las clases que conforman el dominio de un problema, permitiendo identificar y/o reasentar sus propiedades (atributos) y métodos (acciones), para extender su funcionalidad o reusabilidad.

El problema es que desde los años 80 que se inició la investigación en Software Educativo, no se resolvió el divorcio entre los educadores, que sólo se preocupan por los aspectos pedagógicos (motivación, orientación, retroalimentación, evaluación, etc.), y los programadores, que sólo ven la parte funcional. No existe una liga entre ambas. Este trabajo plantea que los patrones de diseño, una idea importante en computación, puede ser tal liga, al definir elementos estructurales que permitan a los programadores crear código que refleje aquello que los educadores pretenden lograr en los productos.

Metodología Empleada.

Para llegar al descubrimiento de los Patrones Educativos de utilizó la siguiente metodología Fig.1

Primero: seleccionamos metodologías de Software Educativo existentes, para identificar sus similitudes, diferencias o en su caso limitantes.

Segundo: después de analizar las metodologías, seleccionamos alguna de ellas con enfoque computacional; así como definiciones de Software Educativo plasmadas en ellas, y de otros autores. El objetivo primordial en esta etapa es identificar posibles asociaciones, clases, atributos, y métodos para cada tipo de software educativo.

*Ricardo Armando Barrera Cámara; docente de tiempo completo en la Universidad Autónoma del Carmen.
José Enrique Álvarez Estrada; director de Redes de Principia Informática, S. A. de C.V., en España.

Tercero: identificados los diversos tipos de Software Educativo sobre la base de sus definiciones y/o características, procedemos a analizar casos de estudio específicos (software existente en el mercado). En esta etapa realizamos declaración del software, descripción del problema, identificación de clases, diccionarios de datos, identificación de asociaciones, diagramas de clase, diagramas de escenario y seguimiento de sucesos para cada caso de estudio.

Cuarto: el siguiente paso es abstraer los patrones de diseño educativo, para lo cual tomamos como base las clases, atributos y métodos claramente identificados en los pasos dos y tres; así como la declaración de patrones de diseño de gamma, uniendo estos elementos nos dan como resultado Declaración de Patrones de Diseño de Software Educativo

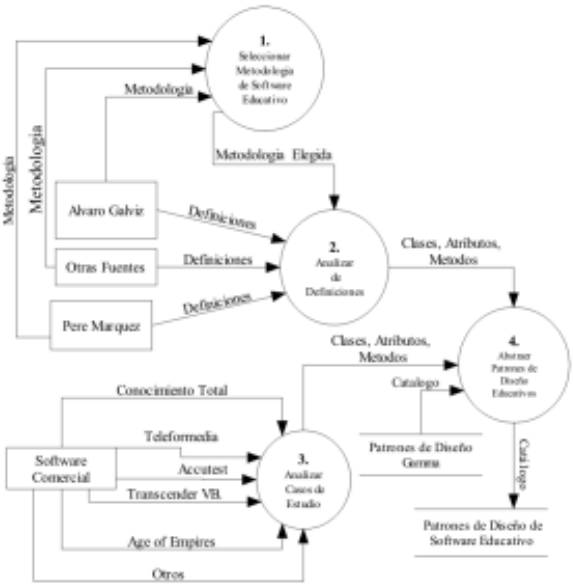

Fig. 1 Diagrama de flujo de datos empleado para el descubrimiento de Patrones de Software Educativos.

Patrones de Diseño.

Un patrón de diseño es una solución reutilizable a un problema que ocurre durante el diseño de software. Cada patrón describe un problema el cual ocurre una y otra vez en nuestro ambiente, y que describe el corazón de la solución, de tal forma que pueda utilizarse esta solución un millón de veces y más, sin tener que repetir lo mismo dos veces.

El propósito de un patrón refleja lo que hace el patrón. Los patrones de creación se refieren a la creación de objetos, los patrones estructurales trabajan sobre la composición de clases y objetos, y los patrones de comportamiento se refieren a cómo las clases y objetos interactúan y distribuyen responsabilidades. El alcance especifica si el patrón es aplicado a clases o a objetos. Los patrones de clase se refieren a la relación entre clases y subclases.

Patrones de Software Educativo

Los patrones que se enlistan a continuación son el resultado de la metodología indicada en la sección dos de este trabajo, para lo cual se utilizo una descripción simplificada de gamma.

Patrón de Software Educativo Ejercitación y Práctica.

Nombre y alcance del patrón. el Patrón de Software Educativo de Ejercitación y Práctica permite representar las clases que conforman un software del tipo ejercitación o práctica (drill & practice).

Intención: el patrón ejercitación y práctica captura la estructura y comportamiento común de un dominio específico, la ejercitación y práctica. Su propósito es facilitar el modelado de este tipo de software mediante su reutilización.

Motivación: la representación en términos de clases y objetos tienen aspectos y elementos comunes que se repiten de un Software de Ejercitación y Práctica a otro, como la interfaz de usuario, ejercicio, cronómetro. Otro aspecto fundamental es la necesidad de diseñarlos e implementarlos.

Estructura de la solución: la estructura, atributos y comportamientos comunes de los Software Ejercitación y Práctica son representados en el siguiente diagrama de clases.

Clases Participantes

Ejercitación y práctica*:* esta clase contenedora principal. Su objetivo es evaluar al estudiante, reuniendo la evaluación particular de cada uno de los ejercicios, almacena la secuencia de ejercicios y decide cuáles presentar y en qué orden (virtual).

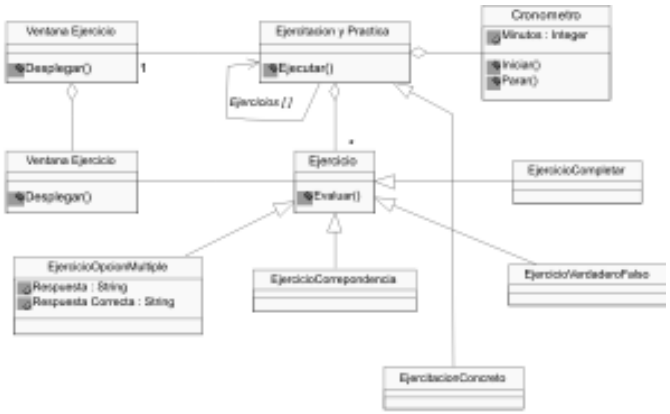

Fig. 2 Diagrama de clases para el Patrón de Software Educativo Ejercitación y Práctica

Ejercicio: almacena una pregunta y/o varias posibles respuestas. Evalúa la respuesta, diciendo si fue correcta o no.

Cronómetro: temporaliza la duración del desarrollo de un ejercicio (tiempo límite o duración)

Ejercicio opción múltiple: genera ejercicios del tipo opción múltiple (la pregunta x tiene sólo como posible respuesta a, b, c, etc.)

Ejercicio correspondencia: generar ejercicios del tipo Ejercicio Opción Correspondencia (la pregunta x le corresponde a, b, etc.)

EjercicioVerdaderoFalso: generar ejercicio del tipo Verdadero/Falso (la pregunta x tiene como respuesta Verdadero o Falso)

Aplicabilidad: la principal aplicación de este patrón en el modelado de Software Educativo de Ejercitación y Práctica.

Consecuencias (Uso): el uso del patrón simplifica el pro-

ceso de modelado y diseño de clases en el contexto de software educativo. El patrón promueve una estructura y comportamiento que puede ser adaptado a las necesidades del usuario. El patrón puede ser reutilizado, en el diseño de software educativo. Al igual que cualquier patrón, el usuario debe adaptarla a los requerimientos de su aplicación, lo cual implica que el usuario puede agregar o modificar nuevos atributos, operaciones, clases.

Usos Potenciales: la generalidad del patrón permite su aplicación en el diseño y desarrollo de software educativo de ejercitación y práctica.

Patrones relacionados: Observer, Bridge. La relación entre vista tutorial y tutorial, así como entre vista ejercicio y ejercicio se modela por el patrón observer. Para emplear múltiples plataformas, se pueden crear distinto descendientes de vista tutorial y vista ejercicio, de acuerdo al patrón Bridge. Para diversificar el tipo de ejercicios posibles, existe una clase ancestro polimorficamente llamada ejercicio con el método virtual evaluar(), de acuerdo nuevamente al patrón Bridge.

Patrón de Software Educativo Tutorial

Nombre y alcance del patrón: el Patrón de Software Educativo de Tutorial permite representar las clases que conforman un software del tipo tutorial.

Intención: el patrón tutorial captura la estructura y comportamiento común de un dominio especifico, el tutorial. Su propósito es facilitar el modelado de este tipo de software mediante su reutilización.

Motivación: la representación, en términos de clases y objetos, tiene aspectos y elementos comunes que se repiten de un software tutorial a otro, como la interfaz de usuario, ejercicio. Otro aspecto fundamental es la necesidad de diseñarlos e implementarlos.

Estructura de la solución: la estructura, atributos y comportamientos comunes del software tutorial son representados en el siguiente diagrama de clases.

Clases participantes

Ejercitación y práctica: e*s*ta clase contenedora principal. Su objetivo es evaluar al estudiante, reuniendo la evaluación particular de cada uno de los ejercicios, almacena la secuencia de ejercicios y decide cuales presentar y en que orden (virtual).

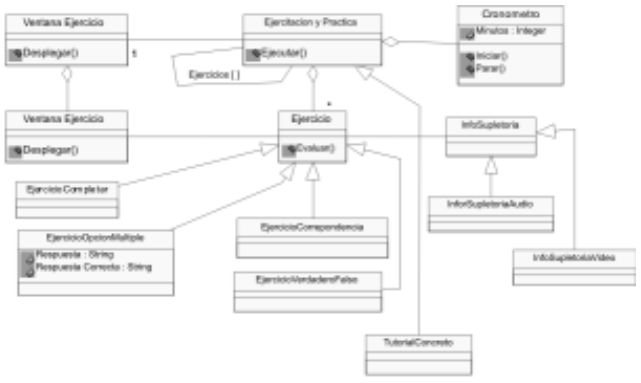

Fig. 3 Diagrama de clases para el Patrón de Software Educativo Tutorial

Ejercicio: almacena una pregunta y/o varias posibles respuestas. Evalúa la respuesta, diciendo si fue correcta o no.

Cronómetro: temporaliza la duración del desarrollo de un ejercicio (tiempo límite o duración)

InformaciónSupletoria: clase que tiene como objetivo proporcionar información a la clase VentanaEjercicio (información adicional del desempeño del usuario) a través de su método desplegar().

EjercicioOpcionMultiple: genera ejercicios del tipo opción múltiple (la pregunta x tiene sólo como posible respuesta a, b, c, etc.).

EjercicioCorrespondencia: generar ejercicio del tipo ejercicio opción correspondencia (la pregunta X le corresponde a, b, etc.)

Ejercicio Verdadero/Falso: generar ejercicio del tipo Verdadero/Falso (la pregunta x tiene como respuesta Verdadero o Falso)

Aplicabilidad: la principal aplicación de este patrón en el Modelado de Software Educativo de Tutorial.

Consecuencias (Uso): el uso del patrón simplifica el proceso de modelado y diseño de clases en el contexto de software educativo. El patrón promueve una estructura y comportamiento que puede ser adaptado a las necesidades del usuario. El patrón puede ser reutilizado en el diseño de software educativo. Al igual que cualquier patrón, el usuario debe adaptarla a los requerimientos de su aplicación, lo cual implica que el usuario puede agregar o modificar nuevos atributos, operaciones, clases.

Usos Potenciales: la generalidad del patrón permite su aplicación en el diseño y desarrollo de software educativo de tutorial.

Patrones relacionados: Observer, Bridge. La relación entre vista tutorial y tutorial, así como entre vista ejercicio y ejercicio, se modela por el patrón observer. Para emplear múltiples plataformas, se pueden crear distinto descendientes de vista tutorial y vista ejercicio, de acuerdo al patrón Bridge. Para diversificar el tipo de ejercicios posibles, existe una clase ancestro polimorficamente llamada ejercicio con el método virtual evaluar( ), de acuerdo nuevamente al patrón Bridge.

Patrón de Software Educativo Simulador y Juego Educativo

Nombre y alcance del patrón: el patrón de Software Educativo Simulador y Juego Educativo permite representar las clases que conforman un software simulador y juego educativo.

Intención: el patrón de Software Educativo Simulador y Juego Educativo captura la estructura y comportamiento común de un dominio específico como es el caso del software simulador y juego. Su propósito es facilitar el modelado de este tipo de software mediante su reutilización.

Motivación: la representación en términos de clase y objetos tienen aspectos y elementos comunes en otras aplicaciones de tipo educativo que pudieran repetirse de un software simulador a otro, o de un juego educativo a otro. Estos a diferencia de los tutoriales, ejercitación y práctica no muestran rasgos que pudieran ser tan obvios, debido a su amplia variedad o diversidad. Pero si podemos abstraer clases como Simulador, Micromundo, Reglas.

Estructura de la solución: la estructura, atributos y com-

portamientos comunes de los software Simulador y Juego Educativo son representados en el siguiente diagrama de clases.

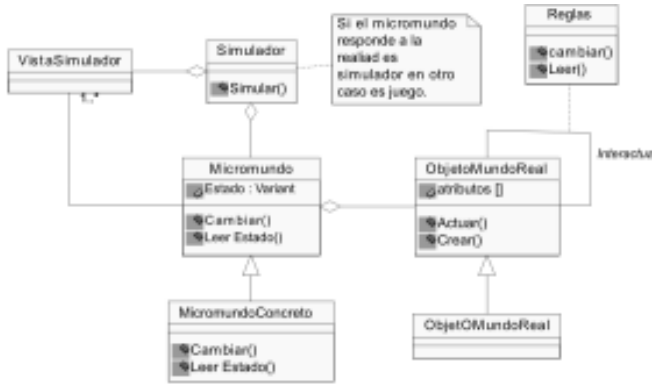

Fig. 4 Diagrama de clases para el Patrón Educativo Simulador y Juego Educativo

Clases Participantes

Micromundo: simula el ambiente controlable por el usuario mediante los métodos cambiar( ) y leer( ), en donde los objetos del mundo real interactúan.

Reglas: también llamada "función de transformación", es una clase que define la interacción ente dos objetos del mundo real.

Simulador: es la clase contenedora a todos los demás, su principal método simular( ).

Objeto mundo real: abstracción de cualquier objeto que son relevante para la simulación o juegos. Todo objeto interactúa con otros objetos a través de reglas.

Aplicabilidad: la principal aplicación de este patrón en el modelado de software educativo de tipo simulador y juego.

Consecuencias (Uso): es uso del patrón, simplifica el proceso de modelado y diseño de clases en el contexto de software educativo. El patrón promueve una estructura y comportamiento que puede ser adaptado a las necesidades del usuario. El patrón puede ser reutilizado, en el diseño de software educativo. Al igual que cualquier patrón, el usuario debe adaptar el patrón a los requerimientos de su aplicación, lo cual implica que el usuario puede agregar o modificar nuevos atributos, operaciones, clases.

Usos potenciales: la abstracción del patrón permite su aplicación en el diseño y desarrollo de Software Educativo de Simulador y Juego.

Patrones relacionados: Observer, Bridge, Builder [3], El patrón Observer define la forma en que interactúan ventana simulador y simulador. El patrón Bridge define la manera en que los descendientes de ObjetoMundoReal lleven a cabo su trabajo con el método actuar( ). El patrón Builder esta presente en la construcción de distintos objetos del mundo real por parte del simulador.

Conclusiones

En la mayoría del software educativo existen patrones de diseño recurrentes que, mediante su estudio detallado y comparación, han sido identificados. No debemos olvidar que los patrones de diseño "no se crean, sino se descubren". Permitiendo una identificación clara y objetiva de estos rasgos como es el caso del Software Tutorial y el Software de Ejercitación y Práctica, no obstante existen software como en el caso de los simuladores y juegos educativos en donde la identificación de patrones de diseño pueda parecer imposible, esto se debe a la gran diversidad y variedad de los mismos. Por tanto, aún queda mucho por descubrir en este campo al identificar patrones más concretos, que describan familias completas de simuladores y micromundos, tal es el caso de los llamados "juegos de estrategia", "juegos de acción", etc.

Por otro lado, el hecho de modelar y diseñar adecuadamente en objetos y clases permite una extensibilidad y reusabilidad del patrón. Los patrones de diseño de software educativo pueden implementarse en lenguajes orientados objetos y no orientados a objetos. Esto facilita la construcción del software, al no obligar al programador a optar por lenguajes o herramientas que desconoce y cuya portabilidad pudiera no ser buena.

Claudia Hernández

Bibliografía

Alexander Christopher, Sara Ishikama, Murray Silverstein. A Pattern Language. Oxford University Press. Nuevo York. 1977

Galvís Panquéva, Álvaro, *Ingeniería de Software Educativo*, Ediciones Uniandes, 1991.

Gamma Erick. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison Wesley Professional Computing Series. 1998.

Gómez Castro Ricardo A. *Ingeniería de Software con Modelaje Orientado a Objetos: Un medio para desarrollar Micromundos Interacrivos.* Informática UNIANDES-LIDIE, Volumen 11,1,1998, pp 9-30.

Larman Graig. *UML y Patrones: Introducción al análisis y diseño orientado a objetos.* Prentice Hall & Pearson. México 1999.

Marqués Pere. *Metodología para la elaboración de Software Educativo. Guía de uso y metodología de diseño.* Barcelona: Editorial Estel 1995.

Rumbaugh James. *Modelado y diseño Orientado a Objetos.* Prentice Hall.

Object Management Group, Inc, Unified Modeling Language(UML).