Informática Educativa
UNIANDES - LIDIE

Vol 11, No, 1, 1998
pp.9-30

INGENIERIA DE SOFTWARE EDUCATIVO CON MODELAJE ORIENTADO POR OBJETOS: UN MEDIO PARA DESARROLLAR MICROMUNDOS INTERACTIVOS

Ricardo A. GÓMEZ CASTRO
Alvaro H. GALVIS PANQUEVA
Olga MARIÑO DREWS

RESUMEN

Las metodologías convencionales de Ingeniería de Software Educativo (ISE) tienen mecanismos robustos para hacer un análisis de necesidades y diseño educativo completos, pero poco han evolucionado con la tecnología en lo relacionado con el diseño computacional. Para hacer uso efectivo de la información recolectada en las fases de análisis y diseño educativo se propone la inclusión del modelo orientado por objetos en todas las etapas del ciclo de desarrollo y así unificar los términos en los que se habla en cada etapa, estableciendo un modelo del mundo del problema y de su comportamiento; de este modo se hace referencia a objetos presentes en el modelo, extendiendo así su funcionalidad. Al llegar a la implementación, los resultados obtenidos se transcriben al lenguaje de programación escogido, cambiando la sintaxis en que se expresa el modelo, mas no la semántica. Esta propuesta se está implementando en LUDOMÁTICA[*], proyecto en el que se circunscribe esta ponencia.

INTRODUCCIÓN

Usar la informática como apoyo a procesos de aprendizaje ha sido una inquietud que durante mucho tiempo ha sido investigada y probada por muchas personas. Su asimilación dentro de instituciones educativas, incluyendo el hogar, ha aumentado en los últimos años, con lo que la demanda por software educativo de alta calidad es cada vez mayor.

* LUDOMÁTICA es un proyecto desarrollado entre la Universidad de los Andes, la Fundación Rafael Pombo y por el Instituto Colombiano de Bienestar Familiar, con la cofinanciación de Colciencias-ETI (Electrónica, Telecomunicaciones e Informática) e ICBF- Subdirección de Protección. Ver detalles en URL http://zeus.uniandes.edu.co/~ludomati

Para lograr software con las condiciones deseadas dentro de las fases de análisis y diseño del mismo se deben incorporar aspectos didácticos y pedagógicos, que faciliten y garanticen la satisfacción de necesidades educativas. Se debe involucrar efectivamente a los usuarios, para conseguir identificar necesidades y/o problemas específicos y se puedan establecer mecanismos de resolución adecuados y apoyar cada una de las fases en sólidos principios educativos y de comunicación humana [i]. Metodologías vigentes de ingeniería de software educativo (ISE) como la propuesta por Galvis [ii] atienden muy bien estos requerimientos y permiten al equipo encargado de dicha labor asumir con propiedad su función.

Por otra parte, la ingeniería de software como disciplina ha evolucionado significativamente en lo que se refiere a modelos conceptuales y herramientas de trabajo [iii, iv, v], que hacen del proceso de desarrollo y mantenimiento de software una actividad cada vez menos dependiente del arte de quienes llevan a la práctica un diseño elaborado. Dentro de estos aportes se destacan los de la orientación por objetos, que cubre todo el ciclo de vida del software [vi]

El objetivo de este trabajo es integrar el modelaje O.O. con la metodología de ISE propuesta por Alvaro Galvis, para enriquecer el proceso de desarrollo de Materiales Educativos Computarizados (MEC) altamente interactivos.

Como punto de partida se identificaron las características que debería poseer un MEC, particularmente un Micromundo Interactivo, fruto de evaluar varias aplicaciones existentes en el mercado con este tipo de micormundos. A partir de allí se hizo adaptación y/o redefinición de los pasos que debe seguir una metodología de ISE en su componente computacional.

MARCO CONCEPTUAL

En esta sección se tratan aspectos relacionados con la Ingeniería de Software (IS) e Ingeniería de Software Educativo (ISE) que serán la base para la integración de la metodología ISE de Galvis [*op.cit*] con las propuestas del paradigma OO.

En la metodología de ISE desarrollada por Alvaro Galvis, los micromundos interactivos[1] juegan un papel clave. Es a través de ellos como se crean ambientes lúdicos para aprender y es en ellos donde se viven experiencias que sirven de base para que el aprendiz

[1] Dice Galvis que "un micromundo es un ambiente de trabajo reducido tan simple o tan complejo como amerite lo que se aprende, donde suceden o pueden suceder cosas relevantes a lo que interesa aprender, dependiendo de lo que el usuario realice. Suele incluir una situación y formas de incidir sobre ella. La situación puede o no ser una fantasía, pero debe evocar algo que sea significante para el aprendiz y que tenga que ver con lo que se va a aprender" [*op.cit.*, pp. **XX**]

genere o apropie conocimiento, dependiendo de la manera (algorítmica o heurística) como se use el micromundo.

UNA METODOLOGÍA DE ISE

El siguiente diagrama ilustra el flujo de acción en la metodología de ISE sobre la que se desea hacer incorporación del enfoque OO. Como se aprecia, el ciclo de vida de una aplicación educativa puede tener dos maneras de ejecución, en función de los resultados de la etapa de análisis: en el sentido de las manecillas del reloj se procede a diseñar, desarrollar y probar lo que se requiere para atender una necesidad. En el sentido contrario, se someta a prueba aquello que se encontró puede satisfacer la necesidad.

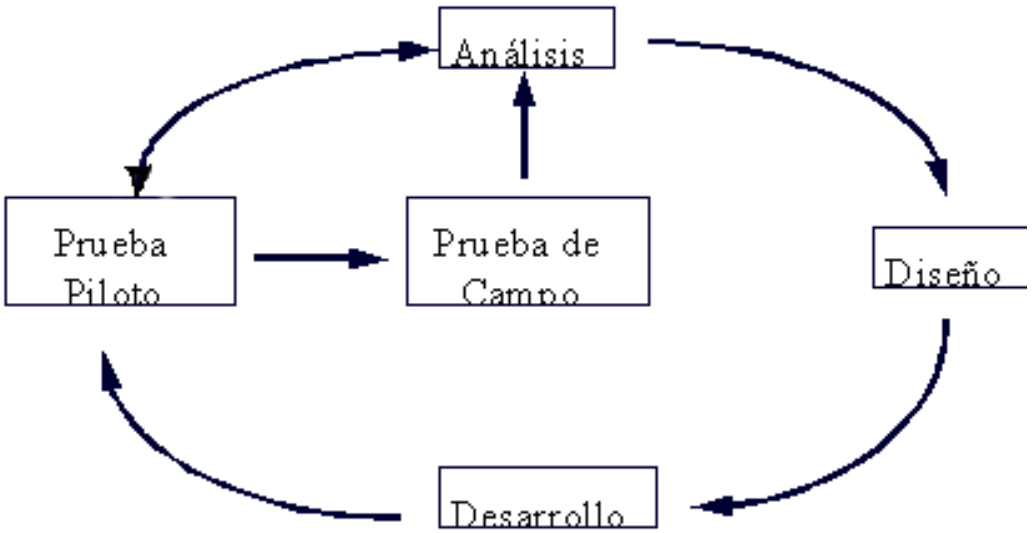


Ilustración 1 - Metodología ISE propuesta por Galvis [2]

La metodología de ISE en mención, publicada en 1991, ofrece mecanismos de análisis, diseño educativo y comunicacional, prueba piloto y de campo bastante sólidos, toda vez que se fundamentan en principios educativos, comunicacionales y de tecnología educativa de validez comprobada. Sin embargo, desde la perspectiva computacional no ha evolucionado, con lo que cabe enriquecerla tomando en cuenta los avances tecnológicos en el diseño y desarrollo computacional que se han logrado en los últimos años. Estos avances permiten incluir dentro de los productos de software nuevos recursos que enriquecen el potencial de acción de los mismos y que cabe usar desde el momento de formular su diseño.

LA INGENIERÍA DE SOFTWARE Y EL PARADIGMA OO

El enfoque de orientación por objetos (O.O.) es un paradigma que también cubre el ciclo de vida del software y que permite tener un mayor acercamiento al mundo que se modela y cómo funciona este mundo. En algunas metodologías de Ingeniería de Software (IS) se habla del análisis, diseño y desarrollo como tres procesos independientes cuya mezcla tiene como resultado final una aplicación que satisface X o Y necesidades. El problema que se presenta esta disyunción está dado por el modo como se trabaja normalmente en cada uno de estos procesos. El lenguaje que manejan, los alcances y el resultado final de cada uno de ellos puede afectar el resultado final global. Al tratarlos como entes independientes, los mecanismos para acomodar y traducir la información producida por cada proceso para que pueda ser "efectivamente usada" genera potenciales fallos a interpretar de determinada manera la información allí contenida.

Mucha información o documentación de apoyo que podría evitar estos problemas no puede ser usada, ya sea porque está incompleta o no existe, o por el contrario existe en demasía o no está "formalizada", generando "ruido" innecesario que en lugar de evitar problemas los acrecienta. Además, cuando un error es detectado, puede que sea una falla que haya estado latente desde el proceso de análisis o diseño y se haya hecho visible en etapas del desarrollo. Es difícil y costoso solucionarlo, porque puede requerir "echar al caño" lo que se ha hecho, e incluso puede ser necesario rediseñar la aplicación.

El enfoque O.O. busca resarcir las deficiencias que se presentan en cada una de las etapas del ciclo de vida de la IS convencional, permitiendo obtener una mejor representación del mundo y de los requerimientos particulares de una aplicación en dicho mundo. Este enfoque puede ser aplicado indistintamente al análisis, diseño o desarrollo de una aplicación. No es estrictamente necesario usar el enfoque en todas las etapas del ciclo de vida de una aplicación. Si se desea, se puede elaborar un buen análisis y diseño O.O., aún cuando la implementación no necesariamente siga el mismo esquema. Sin embargo, es una excelente alternativa usar O.O. en todo el ciclo de vida, buscando aprovechar al máximo todas las bondades de este nuevo paradigma [vii].

CARACTERÍSTICAS DEL ENFOQUE O.O.

Con O.O. se puede hacer representación del mundo que se desea modelar en términos de los objetos que posee. Cada uno de ellos tiene sus propias características que lo identifican y un comportamiento específico. Estos aspectos pueden formalizarse con este enfoque. Con base en las características y comportamiento del objeto se pueden definir invariantes que deben cumplirse, permitiendo así verificar que el objeto funciona como se quiere.

Durante la definición de objetos del mundo se pueden usar los mecanismos de herencia y polimorfismo, para aprovechar las características y comportamiento de algunos objetos

básicos, extendiéndolos para conseguir objetos con un comportamiento más específico[2]. Además se puede usar otra importante característica llamada "reutilización de código", definiendo objetos que pueden ser usados en futuros desarrollos.

VENTAJAS DE USAR EL ENFOQUE OO

Las ventajas de usar el enfoque O.O. se traducen en mejoramientos de calidad a lo largo del ciclo de vida de una aplicación, facilitando además el mantenimiento y la creación de nuevas versiones que extiendan el programa.

Al disminuir las barreras entre las etapas de análisis, diseño y desarrollo, se garantiza que se está hablando de las mismas cosas y en los mismos términos desde el comienzo del análisis hasta el final de la etapa de implementación. Esto evita inconsistencias y permite verificar que las cosas están claramente definidas y cumplen con todos los requerimientos, incluso antes de escribir una línea de código del programa. Las características anteriormente mencionadas (encapsulamiento, herencia, reutilización) permiten crear un software mucho más robusto.

Por último, el hecho de modelar el mundo y no únicamente los datos necesarios para determinada aplicación, permiten crear diversas aplicaciones sobre la misma información sin repetir los procesos de análisis de los mismos. Esto ofrece la posibilidad de dedicarse a cumplir con los requerimientos de la aplicación basándose en las facilidades que ofrecen los objetos del mundo ya modelados.

Se pueden enunciar varios beneficios de la aproximación orientada por objetos [viii]: reutilización de software: permite describir clases y objetos que podrán ser usados en otras aplicaciones; estabilidad: el diseñador piensa en términos de comportamiento de objetos, no en detalles de bajo nivel; diseño rápido y de alta calidad, puesto que se concentra en satisfacer los requerimientos y no en detalles técnicos; integridad; facilidad de programación al usar efectivamente toda la información de la fase de diseño, poniéndola en términos de un lenguaje específico; facilidad de mantenimiento, dado que al tener el modelo del mundo, es fácil realizar mantenimiento en términos de objetos, atributos y métodos de los mismos; independencia en el diseño, el diseño de un software se puede hacer independientemente de plataformas, software y hardware.

LA ISE ENRIQUECIDA CON ENFOQUE OO

En el caso particular de la ISE, usar O.O. en todos los procesos computacionales (análisis, diseño y desarrollo) permite reflejar fácilmente en los ambientes todo aquello que es

[2] Por ejemplo: se tiene definido en el mundo el objeto Persona. Se pueden definir los objetos Empleado y Cliente tomando como base el objeto Persona

importante desde el punto de vista educativo. Esto forma parte del comportamiento del mundo y dicho comportamiento puede ser modelado claramente con este enfoque.

Para poder tener un punto de partida sólido se hizo la identificación de los atributos que cualquier micromundo interactivo debería tener (necesario) y de aquellos que serían opcionales (deseables), revisando diversidad de MECs que son buenos prototipos de software basado en micromundos interactivos.

ELEMENTOS DE UN MICROMUNDO INTERACTIVO

En la siguiente tabla se resumen los elementos que valdría la pena incluir al crear un micromundo interactivo. Esta lista de elementos se hizo tomando como base un grupo representativo de programas existentes en el mercado que incluye micromundos interactivos: TIM[3], Mi castillo de Fantasía[4], Busy Town[5], Mother Goose[6], FAUNA[7], SimCity[8], WarCraft[9], Math Rabbit[10].

Tabla 1 - Elementos de un micromundo Interactivo

Elemento	**Tipo de elemento**
Argumento e historia	Necesario
Variables Compensatorias	Necesario
Variables de Control	Necesario
Variables de Resultado	Necesario
Mundo / Escenarios	Necesario
Retos (Implícitos / explícitos)	Necesario
Personajes y Roles	Necesario
Objetos / Herramientas	Necesario
Zonas de Comunicación	Necesario
Mecanismos de Comunicación Usuario-Aplicación	Necesario
Ambientación / Caracterización	Necesario
Recuperación de estados anteriores	Deseable
Niveles de Dificultad	Deseable

3 *TIM The Incredible Machine* es una producción de Sierra On Line.

4 *Mi Castillo de Fantasía* es una producción de Editorial Anaya.

5 *Busy Town* es una producción de Paramount Interactive.

6 *Mother Goose* es una producción de Sierra On Line.

7 *FAUNA* es una producción de UNIANDES-LIDIE (Laboratorio de I+D sobre Inforática en Educación)

8 *SimCity* es una producción de Maxis.

9 *WarCraft* es una producción de Blizzard Entertainment.

10 *Math Rabbit* es una producción de The Learning Company.

Manejo de información del usuario	Deseable
Mecanisrmos para Análisis de desempeño	Deseable
Ampliación de las posibilidades del micromundo	Deseable
Personalización del ambiente	Deseable
Soporte al trabajo en grupo	Deseable

METODOLOGÍA ISE-OO

La propuesta que se desarrolla en este documento busca unir todo lo anteriormente expuesto - metodología ISE con paradigma O.O - con miras a crear ambientes basados en micromundos interactivos. El gran reto es diseñar e implementar *micromundos altamente interactivos* que tomen muy en cuenta el potencial tecnológico y los recursos disponibles actualmente, sobre una *sólida base educativa y comunicacional*.

El enfoque base para la conceptualización y diseño de micromundos está desarrollado en el libro de Galvis [*op.cit,* caps. 6 y 7], y las adiciones propuestas provienen de mecanismos de ingeniería de software usados actualmente en Ludomática para el análisis y diseño de MECs [ix]. Para establecer la estructura genérica sobre la cual se puedan "montar" micromundos lúdicos se va a tener en cuenta el conjunto de elementos mencionados en Galvis [*op.cit.*] y se usa el enfoque O.O. para definir el modelo de datos. La Notación usada en este modelaje modelo es UML [x].

Siguiendo el ciclo de vida de un MEC, la siguiente descripción permte entender cada una de sus etapas, enriquecidas con el enfoque OO mencionado.

ANÁLISIS

El objetivo de esta etapa es determinar el contexto en el cual se va a crear la aplicación y derivar de allí los requerimientos que deberá atender la solución interactiva, como complemento a otras soluciones basadas en uso de otros medios (personales, impresos, audio-visuales, experienciales), teniendo claro el rol de cada uno de los medios educativos seleccionados y la viabilidad de usarlos.

De acuerdo con Galvis [*op.cit*] en esta etapa se establece como mínimo la siguiente información:

- *Características de la población objetivo*: edad (física y mental), sexo, características físicas, y mentales (si son relevantes), experiencias previas, expectativas, actitudes, aptitudes, intereses o motivadores por aprender.

- *Conducta de entrada y campo vital*: nivel escolar, desarrollo mental, físico o psicológico, entorno familiar y escolar, etc.
- *Problema o necesidad a atender*. Para establecer la necesidad se puede recurrir a los mecanismos de análisis de necesidades educativas en [2, cap. 5]. Estos mecanismos usan entrevistas, análisis de resultados académicos, etc. para detectar los problemas o posibles necesidades que deben ser atendidas. El problema o necesidad no tiene que estar necesariamente relacionado con el sistema educativo formal, pueden ser necesidades sentidas, económicas, sociales, normativas, etc.
- Una vez identificado el problema se deben establecer las bases para resolverlo. *Principios pedagógicos y didácticos aplicables* [2, cap. 4]. En esta fase se debe analizar cómo se ha llevado a cabo el proceso de enseñanza-aprendizaje para establecer cómo debe enfocarse el ambiente, qué factores tomar en cuenta, qué objetivos debe cumplir.
- *Justificación de uso de los medios interactivos* como alternativa de solución. Para cada problema o necesidad encontrada se debe establecer una estrategia de solución contemplando diferentes posibilidades. El apoyo informático debe ser tomado en cuenta siempre y cuando no exista un mecanismo mejor para resolver el problema: soluciones administrativas, ver si el problema se soluciona al tomar decisiones de tipo administrativo; soluciones académicas, cambios en metodologías de clase; mejoras a los medios y materiales de enseñanza contemplando el uso de medios informáticos. Una vez que se han analizado todas las alternativas se puede decir por qué el uso de medios informáticos es una buena solución. La justificación se puede basar en la no existencia de otro medio mejor y en la relación costo-beneficio para la institución pues puede ser que exista una mejor solución pero que demande mayor tiempo y esfuerzo o un mayor costo económico, etc.

ESPECIFICACIÓN DE REQUERIMIENTOS

Como síntesis de la etapa de análisis se deben formular los requerimientos que deberá atender el material interactivo que se desea obtener.

La especificación de requerimientos debe contener lo siguiente:

- *Descripción de la Aplicación*: Contiene las características particulares de la aplicación dentro de determinado dominio: área de contenido, restricciones etc. Se hace una descripción de lo que hará la aplicación.
- Además se deben dejar claras las restricciones que tendrá y una descripción de los posibles escenarios de interacción que tendrá el usuario.

➢ Las restricciones están relacionadas con aspectos tales como:

Población Objetivo y sus características (información recopilada en la fase de análisis).

Areas de contenido y sus características.

Principios pedagógicos aplicables

Modos de uso de la aplicación: individual, grupal, con apoyo de instructor, etc.

Conducta de entrada. Todo aquello con lo que el usuario cuenta antes de usar la aplicación: experiencias, conocimiento, habilidades, etc.

- Los escenarios de interacción corresponden a los momento de interacción que tendrá el usuario en cada uno de los ambientes del mundo. Por ejemplo, el registro de datos al iniciar la aplicación, la escogencia de herramientas, etc.

- *Diagramas de Interacción*: Permiten ver secuencias de interacción entre el usuario y la aplicación, representando lo que se espera del diálogo y dando más detalle a la descripción textual de la descripción de la aplicación. Los diagramas de interacción son un formalismo que permite ver la secuencia de acciones entre diferentes partes de la aplicación involucrada en llevar a cabo determinada actividad. Es importante ver la secuencia de acciones para cada escenario de interacción. Con base en estos diagramas se pueden ver cuáles pueden ser las necesidades de información en cada escenario de interacción y se puede ir pensando en cuáles pueden ser los algoritmos que serán usados.

Los diagramas de interacción mencionados en esta etapa tienen la siguiente sintaxis:

<NOMBRE DIAGRAMA>

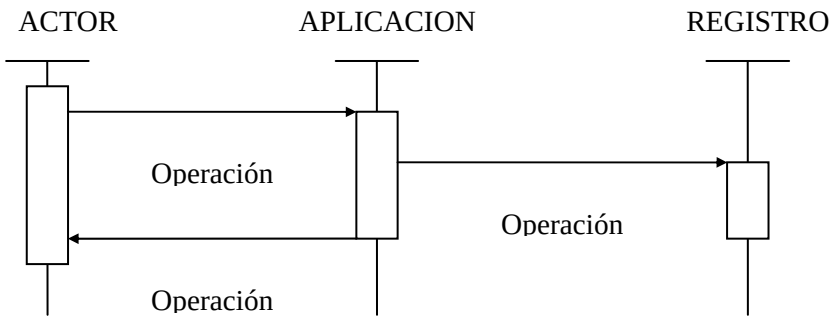


Ilustración 2 - Diagrama de Interacción

El actor en este caso corresponde a cada uno de los diferentes usuarios de la aplicación. Los objetos, aplicación y registro corresponden en este caso a las partes de la aplicación involucradas en el diagrama. Nótese que en este momento no se habla de ningún modelo de datos específico, simplemente se especifica qué es lo que va a hacer la aplicación.

Las operaciones que aparecen en el diagrama son requerimientos de información que se comparten entre cada uno de los diferentes objetos. Con base en estas operaciones se puede especificar la secuencia para llevar a cabo la acción objetivo del diagrama. Se debe tener un diagrama por cada escenario de interacción de la aplicación

DISEÑO

El diseño del Micromundo Interactivo se realiza a tres niveles diferentes: educativo, comunicacional y computacional. La metodología de ISE original es fuertes en cuanto al diseño educativo y diseño comunicacional de MECs. En esta propuesta ISE-OO se van a tomar en cuenta estas fortalezas y se van a usar de manera que sean reflejadas en el diseño computacional de la aplicación y en la implementación de la misma.

Al diseñar el ambiente en el que se desarrollará la acción se deben definir claramente los elementos que se determinaron como necesarios en todo micromundo interactivo y aquellos deseables que convenga para el caso (ver Tabla 1). La identificación de estos elementos en esta etapa permite crear mayor vínculo con la etapa de desarrollo. Muchas de las decisiones importantes acerca del micromundo y su comportamiento se toman aquí.

Se va a realizar el diseño usando el enfoque O.O., formalizando muchos de los aspectos relacionados con la aplicación, definiendo desde esta etapa los objetos, su comportamiento, el propósito de la aplicación, las restricciones existentes y los escenarios de interacción.

Como complemento al diseño educativo de ISE, se plantea el uso de una metodología de especificación y diseño que acerque mucho más los resultados y formulaciones hechas en dicho diseño educativo hacia la implementación de la aplicación. Con esto se está garantizando un diseño computacional y posterior implementación con una alta calidad. Cualquier ajuste se puede hacer en etapa de diseño, reduciendo costos innecesarios en etapa de desarrollo.

En este documento se usa como base una propuesta planteada en por Figueroa [op.cit y xi], la cual servirá como soporte al diseño O.O. y posterior diseño de datos e implementación de la aplicación. Se va usar UML [9] para la notación del modelo. Se desea así obtener una arquitectura genérica para micromundos interactivos, que pueda extenderse para satisfacer necesidades de un problema en particular. Junto con la arquitectura se debe especificar la funcionalidad que el usuario tendrá sobre el modelo, para saber qué cosas puede hacer sobre él.

A continuación se define cada una de las etapas del diseño: diseño educativo, diseño comunicacional. diseño computacional.

DISEÑO EDUCATIVO

Tomando como punto de partida la necesidad o problema, así como la conducta de entrada y campo vital de la población objeto, se debe establecer lo que hay que enseñar o reforzar para

subsanar con apoyo del MEC las necesidades encontradas. Como resultado de la fase de diseño educativo se debe tener lo siguiente: contenido y su estructura; micromundo; sistema de motivación; sistema de evaluación. De acuerdo con Galvis [op.cit, cap 6] el diseño educativo debe resolver los siguientes interrogantes: ¿Qué aprender con el MEC? ¿En qué micromundo aprenderlo? ¿Cómo motivar y mantener motivados a los usuarios? ¿Cómo saber que el aprendizaje se está logrando? Las relaciones entre estos elementos se pueden visualizar así:

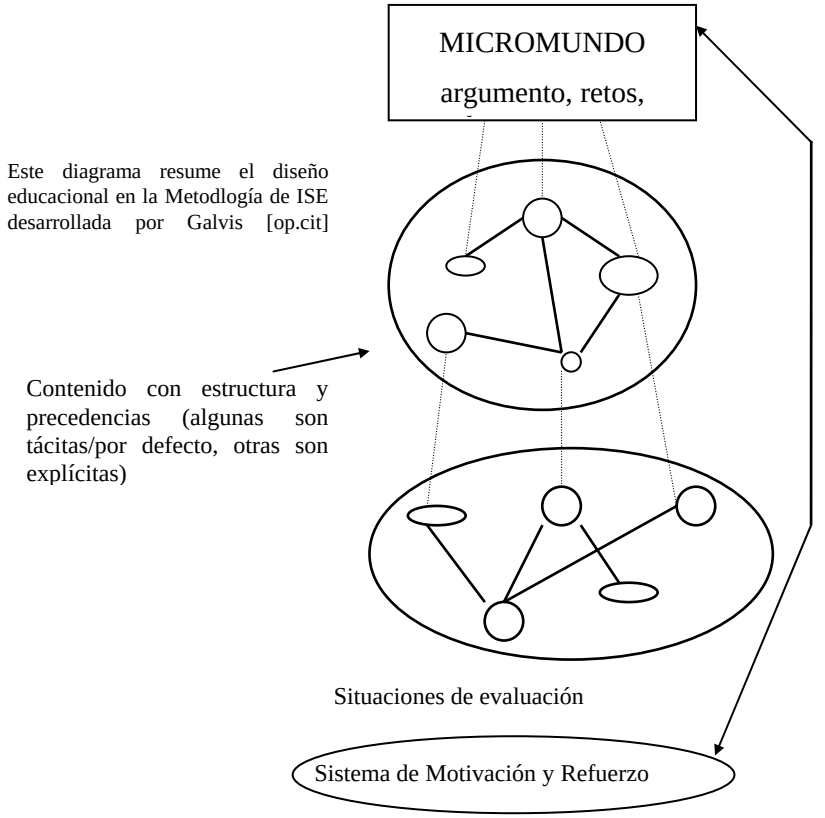


Ilustración 3 - Diseño educativo de un MEC

Qué aprender con el MEC ?

Para resolver este interrogante se debe partir del qué que subyace a micromundo: contenidos a tratar, derivados de las necesidades o problemas, tratando de detallar las unidades de contenido que van a tomarse en cuenta en el MEC. Se debe definir la red semántica que relaciona los conceptos que interesa desarrollar en la aplicación. Con base en esta red se puede establecer la

base de datos de contenidos que soporta el material. Debe cuidarse la manera como se presentan los contenidos en el MEC. Las relaciones de dependencia entre los diferentes temas deben tomarse en cuenta para no forzar el paso de un tema a otro y mantener coherencia a lo largo del material.

Se debe tener clara la diferencia entre lo que se sabe antes de usar el MEC y lo que se espera que se sepa al finalizar el trabajo con éste: Objetivos, contenidos y sus interrelaciones. Siguiendo la idea de Galvis [*op.cit*, cap. 6] se debe establecer esto en términos operacionales, estableciendo los contenidos a tratar y el objetivo terminal del MEC [*op.cit*, cap.13] y luego descomponiendo éste en objetivos específicos y secuenciándolos.

En qué ambiente o micromundo aprenderlo ?

Un MEC se compone de varios ambientes o micromundos, cada uno relacionado con un objetivo en particular. Para cada micromundo se debe establecer: Argumento, Mundo, Escenarios, Retos, Personajes y Herramientas, Objetos. Siguiendo el modelo O.O., se deben definir las *clases* (ver glosario) que identifican cada uno de estos elementos. Algunas de estas clases serán la base sobre la cual se puede extender el micromundo. Al realizar el modelaje del mundo se deben definir las relaciones existentes entre estas clases.

La definición de los elementos del micromundo (escenarios, objetos, etc.) se expresa usando una tabla como la siguiente:

Tabla 2 - Especificación general de los elementos del Micromundo Interactivo

ELEMENTO	CARACTERÍSTICAS	QUÉ SE PUEDE HACER CON EL ELEMENTO ?
Nombre del elemento Observaciones	Información que se desea tener en el elemento	Qué necesidades de información satisface el elemento ?

Una tabla como ésta permite hacer una clasificación inicial de todo lo que está en el mundo que se está modelando. Además, al tener claras las características y lo que se puede hacer con cada elemento del mundo, se pueden establecer relaciones entre ellos. Estos elementos son posibles clases de objetos; al refinar su definición y al establecer las relaciones se puede saber cuáles de estos elementos serán clases que harán parte del modelo estático del mundo y cuales son simplemente atributos complejos de alguna clase de dicho modelo.

Además se debe definir qué cosas puede hacer el usuario en el mundo. En términos de UML se refiere a los *casos de uso* en el mundo. Los casos de uso se identifican al establecer los requerimientos de información que debe satisfacer la aplicación. Los casos de uso pueden extenderse de acuerdo con las necesidades del problema. Cada caso de uso se especifica usando diagramas de interacción que permitan ver los objetos que están involucrados así como la secuencia de mensajes entre ellos.

Cómo motivar y mantener motivados a los usuarios ?

Según Mockus [xii] Seymour Papert cree que una de las contribuciones principales de Piaget, más allá del concepto de estadios de desarrollo, es mostrar que la gente posee diferentes teorías acerca del mundo. De acuerdo con esto, los niños aprenden mejor cuando son alentados a apoyarse sobre su propia intuición y a emplear lo que ya saben para desarrollar nuevas ideas.

En esta etapa del proceso de diseño se definen las metáforas usadas, así como cada personaje que aparece, dejando claro cuál es el rol que el usuario juega., las herramientas de interacción que podrá usar y cuál es el reto que debe resolver.

En el caso de los micromundos interactivos es vital despertar motivación intrínseca proponiendo ambientes o situaciones que sean interesantes, que despierten curiosidad, que inviten al usuario a indagar a través de la experimentación con el micromundo. Hay que mantener motivados a los usuarios para que el trabajo que se tenga con la aplicación sea efectivo y de provecho. El micromundo debe ser novedoso y buscar sorprender al usuario, darle nuevas oportunidades de acción y plantear nuevos retos. Esto aumenta la curiosidad de los usuarios y los mantiene atentos al desarrollo del trabajo con la aplicación. Complementariamente se deben plantear retos que mantengan alerta a usuario en busca de pistas para resolverlos y con un nivel de complejidad apropiado.

El uso de ambientes educativos debe propiciar la generación de motivación intrínseca en los usuarios, para lograr un efecto duradero en el proceso de enseñanza aprendizaje. Además el uso de fantasías que sean interesantes para ellos, para llegar a lo que Piaget llama intento de asimilar experiencia en las estructuras existentes en su mente, con mínimas necesidades de acomodarlas a las demandas de una realidad externa [xiii]. Es por eso que personas como Richard Pattis [xiv] han coincidido en la necesidad de crear ambientes educativos que aprovechen la motivación que los niños sienten por usar juegos de vídeo como Nintendo, antes de forzarlos a usar esquemas tradicionales de aprendizaje, p. e. libros.

La especificación unida a los resultados del diseño educativo puede ser usada como información de base para usar herramientas como las encontradas en Rational [*op.cit*] para elaborar el diseño O.O. de los datos del mundo de la aplicación. Hay que reflejar la motivación en el modelo. Esto se nota adicionando eventos al modelo así como estableciendo relaciones para que las clases del modelo reaccionen acorde con todo lo que pudiera suceder en el modelo. Estas reacciones lograrían captar la atención del usuario e incluso generar mayor curiosidad ante el comportamiento de la aplicación ante las acciones y decisiones que se tomen.

Cómo saber que el aprendizaje se está logrando ?

Las situaciones de evaluación (retos, etc.) deben estar relacionadas con los contenidos. La relevancia y pertinencia de determinado reto o prueba se debe sustentar con base en los contenidos que se han presentado y con la manera como han sido tratados.

Situaciones de evaluación

El sistema de evaluación está relacionado con todos los retos del mundo. De acuerdo con esto debe definirse el nivel de logro para cada reto, que unido con todas las características (nivel de dificultad, tipo de aprendizaje, etc.) debe permitir evaluar qué ha hecho el usuario en el mundo y si lo hizo correctamente o no. Estos indicadores de logro deben llevarse en la historia que el usuario tiene.

Hay que tener en cuenta el tipo de cosas que se desea aprender: si el aprendizaje es reproductivo, si es de nivel superior o si lo que se aprende es afectivo o psicomotor.

En función del momento de evaluación existen varios tipos de evaluación para usar: evaluación sumativa: averiguar cuánto logró el aprendiz; evaluación diagnóstica: aplicada antes de iniciar la interacción con el MEC, para saber el punto de partida; evaluación formativa: situaciones para ayudar a descubrir o practicar, transferir y afianzar destrezas, conceptos o habilidades.

Los retos que se presentarán al usuario se deben establecer de acuerdo con el contenido: descripción, representación gráfica (si es aplicable) y solución (o mecanismo de verificación para retos más complejos).

Tabla 3 - Definición de retos en el MEC

OBJETIVO	TIPO DE RETO	OBSERVACIONES
Aquí se relacionan los objetivos del MEC	Para cada objetivo se define el tipo de restos que pueden usarse	Para cada tipo de reto se debe especificar cómo es la descripción, representación, tipo de pistas y modo de obtenerlas (si se van a dar) y mecanismos de solución

Manejo de retroinformación, refuerzo y niveles de logro

Dependen mucho del enfoque del micromundo, según sea para aprendizaje por descubrimiento (enfoque heurístico) o por transmisión (enfoque algorítmico). En el caso de ambientes heurísticos como es el caso de la mayoría de los micromundos interactivos, la retroinformación se traduce en mostrar en el micromundo el efecto de lo que hizo el usuario, independientemente de si es correcto o no, para que éste sea quien analice lo que ha pasado y tome decisiones al respecto.

Para decidir si el usuario ha logrado determinado nivel de aprendizaje se deben establecer criterios claros. Para esto deben extenderse todos los elementos del modelo computacional del mundo para que reaccionen ante determinados eventos. Estos eventos deben modelarse especificando qué eventos genera cada elemento del modelo (mundo, escenario, etc.). Además se debe especificar para cada elemento del modelo ante cuáles eventos está en capacidad de reaccionar. Esto se puede lograr definiendo una clase Evento a partir de la cual se pueden establecer todos los eventos del sistema. Esta clase estaría relacionada con todos los elementos del modelo que deseen generar un tipo de evento que identifique acciones hechas por el.

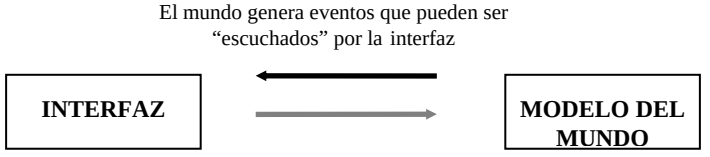
DISEÑO COMUNICACIONAL

En esta fase del proceso de diseño se define la interfaz (zona de comunicación usuario-programa) de la aplicación. En este momento se debe complementar ese bosquejo definiendo formalmente los objetos que posee cada pantalla y cuáles elementos del mundo son usados/afectados. Se toma como base la descripción macro dada en especificación. Es importante conseguir que la interfaz sea: amigable, flexible y agradable de usar; también debe ser consistente, es decir, cuidando que los mensajes y la distribución en pantalla, el juego de colores, etc. sigan un mismo patrón, también es necesario que sea altamente interactiva, lo cual conlleva tener mecanismos de comunicación entre el usuario y la aplicación.

Al definir la interfaz se debe tener en cuenta: ¿cuáles dispositivos de entrada-salida conviene poner a disposición del usuario para trabajar con el Micromundo ?, ¿qué zonas de comunicación entre usuario y programa debe tener el Micromundo ?, ¿cuáles son las características de dichas zonas de comunicación ?, ¿cómo verificar que la interfaz satisface los requisitos mínimos deseados ?. Para cada pantalla de la interfaz se deben definir las zonas de comunicación así como la distribución de las mismas. Para hacer esto se deben seguir indicaciones de diseño de interfaces. En Galvis [*op.cit*, cap.7] se hace una revisión de aspectos a tomar en cuenta.

Al diseñar una interfaz también se deben tomar en cuenta restricciones tecnológicas, características de la población y aspectos psicológicos de la percepción [*ibid*].

Así como se estableció un modelo para el mundo, se debe establecer un modelo para la interfaz que esté atento a todo lo que ocurre en el mundo pero que sea independiente de él. El esquema de interacción entre el mundo y la interfaz se muestra en el siguiente diagrama:



La interfaz se comunica con el mundo usando los mensajes provistos por el modelo

Ilustración 4 - Interacción Interfaz-Modelo del Mundo

El modelo computacional de la interfaz consta de:

- Definición formal de cada pantalla
- Objetivo
- Eventos del modelo del mundo que está en capacidad de detectar
- Diagrama de la pantalla, indicando cuáles objetos tiene y dónde están ubicados.
- Listado de las características tanto de la pantalla como de cada objeto (colores, tamaño de fuentes, resolución de imágenes, etc.)
- Enlaces con otros elementos de la interfaz. En caso de que algún objeto (p. ej. botones) permitan "viajar" a otras pantallas.
- Notas adicionales. En caso de que se requiera realizar operaciones especiales en la interfaz. Por ejemplo indicar si hay animación cuando se activa o desactiva la pantalla, si hay música de fondo, etc.
- Diagrama de flujo de información en la Interfaz. te diagrama indica la relación entre las diferentes pantalla de la interfaz. Con este diagrama se puede establecer cual es la secuencia que se seguirá en la aplicación.

DISEÑO COMPUTACIONAL

Al final de esta etapa se tiene como resultado, claramente definidas, cada una de las diferentes clases de objetos, incluyendo sus atributos (indicando si serán públicos -visibles a todo el mundo- o privados), el conjunto de métodos y el invariante de cada clase que corresponde al conjunto de restricciones o de requisitos que debe siempre cumplir una determinada clase. Por ejemplo, se puede tener definida una clase "reloj" que tiene como atributo un intervalo de tiempo. El invariante de esta clase puede ser tan sencillo como "el intervalo debe ser siempre mayor o igual a cero".

Durante las fases de diseño educativo y comunicacional se han definido los diferentes objetos tanto del mundo como de la interfaz. Esta información se refina en esta fase, adecuándola a las posibilidades de la herramienta de desarrollo que se vaya a utilizar. Algunas clases necesitarán extenderse para ser usadas en el modelo.

Además se puede dar el caso de agregar nuevas clases y relaciones al modelo para dar mayor funcionalidad al modelo acorde con los requerimientos propios de la aplicación. La herramienta de desarrollo puede ofrecer mecanismos que faciliten la implementación de las interfaz. En caso de no ser así, el modelo del mundo se extiende de tal manera que pueda comunicarse efectivamente con el modelo de interfaz que deberá ser desarrollado.

Junto al conjunto de clases, llamado también *modelo estático del mundo*, se debe ilustrar la lógica acerca de cómo se desarrollan cada una de las actividades en el modelo. Para ello se deben refinar los casos de uso (algunos de los cuales ya se han obtenido en fases anteriores, ilustrando para cada uno de ellos el proceso que se sigue. Para hacer esto se pueden usar diagramas de interacción que pueden ser de dos tipos: diagramas de secuencia (similares a los usados en la fase de especificación) o diagramas de colaboración. En estos diagramas ya se puede ver la secuencia de mensajes entre los diferentes objetos involucrados en cada caso de uso y se pueden modelar todas las alternativas que puedan presentarse en cada caso.

Esta información puede ayudar a redefinir el modelo antes de iniciar la fase de desarrollo. Además permite validar si el modelo es completo y permite satisfacer todos los requerimientos de la aplicación.

La ilustración 5 muestra los casos de uso generales de una aplicación que atiende la funcionalidad de micromundos interactivos. Estos casos de uso corresponden a aquellos que son satisfechos en el modelo genérico del mundo (ver Jacobson [*op.cit*]). Estas son las cosas básicas que puede hacer el usuario: puede recorrer todos los escenarios del mundo y en cada uno de ellos resolver retos. Puede interactuar con personajes y así obtener pistas para resolver determinado reto. Además puede recoger objetos que encuentra a su paso e incluso usar herramientas para afectar el escenario.

La ilustración 6 muestra el modelo de clases de mundo para un micromundo interactivo. Este modelo puede considerarse como la base sobre la cual se pueden montar todos los elementos presentes en la aplicación. Este modelo usa notación UML. En dicho modelo se tiene el mundo y su conjunto de ambientes. Cada ambiente o escenario tiene un conjunto de objetos, herramientas, retos y personajes. El usuario puede navegar por el mundo libremente, cambiando de escenarios, resolviendo retos e interactuando con personajes.

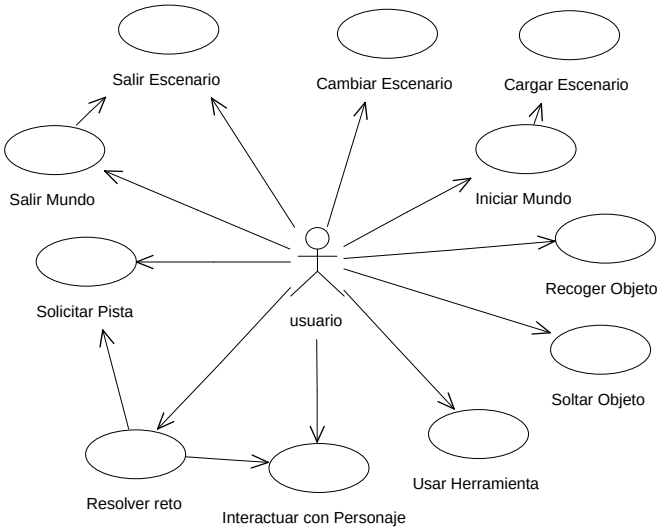


Ilustración 5 - Diagrama de casos de uso para un micromundo interactivo

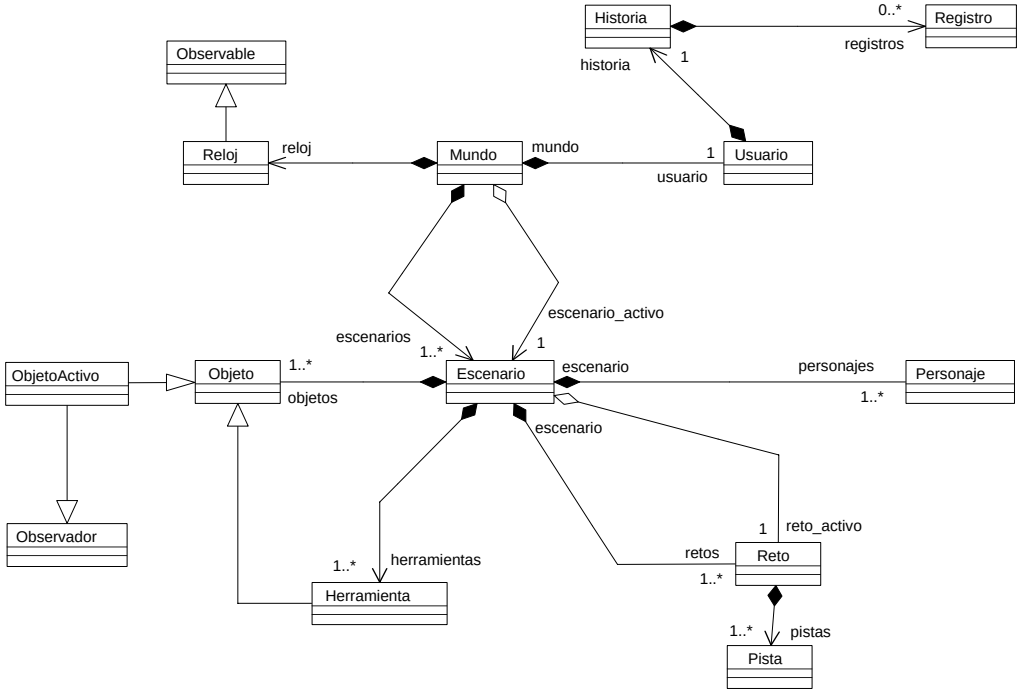


Ilustración 6 - Modelo UML del Mundo, para un micromundo interactivo

Hay que estar atento a cuanto sucede en el modelo del micromundo. Para esto deben extenderse todos los elementos del mundo para que reaccionen ante determinados eventos. Estos eventos deben modelarse especificando qué eventos genera cada elemento del modelo (mundo, escenario, etc.). Además se debe especificar para cada elemento del modelo ante cuáles eventos está en capacidad de reaccionar.

Para ello se puede definir una *clase Evento* a partir de la cual se pueden establecer todos los eventos del sistema. Esta clase está relacionada con todos los elementos del modelo que deseen generar un tipo de evento que identifique acciones hechas por el.

Dentro de los eventos que generan las clases del modelo están:

Tabla 4 - Eventos en el Modelo

ELEMENTO	EVENTOS
Mundo	Iniciar Aplicación Terminar Aplicación Cambio de escenario
Escenario	Cambio de Reto Resolución de Reto Recoger Objeto Soltar Objeto
Usuario	Actualizar de la Historia Escoger objeto Activar objeto
Personaje	Hablar con el usuario Dar objetos al usuario
Herramienta	Activar herramienta Desactivar herramienta

Estos eventos pueden aumentar de acuerdo con el sistema de motivación y las relaciones existentes entre personajes, herramientas y cosas específicas dentro del argumento del micromundo. Para poder “escuchar” eventos en el sistema se debe tener *una clase Escucha*.

El modus operandi de la relación Evento-Escucha es similar a la del modelo observador-observado mencionado anteriormente. Se define un clase evento específica para cada clase del modelo que desee manejar eventos. Esta clase se encarga de despachar solamente los eventos relacionados con una clase en particular.

Además se debe crear una clase Escucha para cada tipo de eventos del modelo (eventos de mundo, eventos de escenario, etc.) Cada clase Escucha está atenta a recibir solamente los eventos de determinada clase.

DESARROLLO

En esta fase se implementa la aplicación usando toda la información obtenida anteriormente. Se toma la definición de clases y se implementa en el lenguaje escogido (Java, Delphi...), tomando en cuenta las restricciones computacionales que se tengan. Hay que establecer la herramienta de desarrollo sobre la cual se va a implementar la aplicación. Los criterios para escogerla incluyen; costo, disponibilidad en el mercado, portabilidad de la aplicación desarrollada, facilidades al desarrollador (ambientes gráficos de desarrollo, mecanismos de depuración, manejo de versiones, etc.).

En el desarrollo se busca que el modelo del mundo sea independiente de la interfaz. Esto facilita el trabajo y permite trabajar en paralelo.

La interfaz se implementa usando la especificación del diseño comunicacional. En algunos ambientes de desarrollo la creación de ésta se facilita con herramientas visuales de desarrollo. En otros se tiene que programar cada uno de los elementos de la interfaz.

PRUEBA A LO LARGO Y AL FINAL DEL DESARROLLO

La metododología propuesta permite ir depurando los componentes del modelo generado, haciendo validacuón con expertos de los prototipos durante la etapa de diseño y prueba uno a uno de los módulos desarrollados, a medida que estos están funcionales.

Superada la depuración y ajuste, se pone a disposición una versión beta del micromundo interactivo. Esto conviene hacerlo con una muestra de la población; se pretende a través de dicha prueba piloto verificar que efectivamente la aplicación satisface las necesidades y cumple fon la funcionalidad requerida.

CONCLUSIONES

El desarrollo de micromundos interactivos es una necesidad actual que debe ser atacada por desarrolladores de software educativo. El avance tecnológico unido con la cultura informática cada vez mayor a nivel de estudiantes y profesores, permite pensar en tener materiales educativos computarizados cada vez más sofisticados que exploten todo el potencial tecnológico en pro de apoyar efectivamente el proceso de enseñanza-aprendizaje.

La inclusión del modelo O.O. articulado al ciclo de ISE permite aprovechar todo el potencial de las metodologías de ISE y de la moderna IS-OO. Esto es importante a la hora de desarrollar software de calidad. Esta integración de enfoques facilita el mantenimiento computacional del mundo en el que se desarrolla la acción, así como la expansión de éste a medida que se requiera, garantizándose así integridad con cada cambio que se realice en el modelo del mundo.

El esquema de interacción entre la interfaz y el modelo del mundo permite trabajar en paralelo cada uno de ellos y permite realizar cambios sin afectar el proceso de desarrollo. Por otra parte, al trabajar O.O. se facilita la reutilización de código así como la portabilidad del mismo en el caso de usar lenguajes de programación O.O. como JAVA.

La metodología que se propone permite montar un micromundo interactivo sobre una estructura genérica ya desarrollada, extendiéndola para satisfacer requerimientos específicos.

GLOSARIO DE TÉRMINOS

- *Atributo*: Es cada una de las características de un objeto: identificador, descripción...
- *Caso de uso*: Corresponde a cada cosa que puede hacer un usuario dentro del modelo de datos. La identificación de estos casos de uso se hace con base en los requerimientos de la aplicación a desarrollar
- *Clase*: Definición de atributos y métodos para un conjunto de objetos.
- *Depuración*: Hace referencia a métodos para refinar el código del programa que se está desarrollando, identificando y eliminando todos los posibles errores que éste tenga.
- *Diagrama de interacción*: Indica las secuencia de acciones que deben seguirse para realizar una tarea en el modelo computacional. Este tipo de diagrama puede indicarse de dos maneras: diagrama de secuencia, en el cual se muestra la secuencia lineal de acciones en determinado momento; diagrama de colaboración, muestra la secuencia de acciones de modo no lineal, resaltando las relaciones y/o dependencias entre diferentes clases del modelo..
- *Escenario de interacción*: Cada uno de los momentos de interacción que tiene el usuario con la aplicación (registrarse, escoger reto, etc.).
- *Invariante de clase*: Es todo aquello que debe cumplir siempre cada clase. Por ejemplo, si se definiera la "clase Persona" se tendría el siguiente invariante: "*una Persona siempre tiene nombre, apellido y documento de identidad. No existen dos personas con el mismo documento de identidad*".
- *JAVA*: Es un lenguaje de programación O.O. desarrollado por Sun Microsystems.
- *Método*: Corresponde a cada una de las funciones que puede llevar a cabo un objeto, p.ej. crearse, destruirse, dar su identificación, etc.
- *Modelo estático*: En la notación UML, el modelo estático corresponde al diagrama donde se muestran todas las clases definidas para la aplicación, indicando para cada clase sus atributos y métodos, así como las relaciones que tiene con las demás clases.
- *Modelo dinámico*: Corresponde al con junto de casos de uso de la aplicación.
- *Objeto*: En el enfoque O.O. un objeto es cualquier cosa que puede ser identificada plenamente en el mundo, es decir que tiene unas características y comportamiento particulares.
- *Polimorfismo*: Es una característica presente en la programación
- *Portabilidad***:** Capacidad que tiene una aplicación de ejecutarse en diferentes plataformas de hardware y software.

- *Prueba piloto:* Prueba de la aplicación realizada con un grupo representativo de la población objetivo
- *UML*: Unified Modeling Language. Es una manera estándar de modelar los datos de determinada aplicación, con una notación para expresar los datos (atributos, métodos), las relaciones entre los mismos y el con junto de requerimientos que pueden ser satisfechos en la aplicación.

AGRADECIMIENTOS

Este trabajo se realizó en el proyecto LUDOMÁTICA, el cual es una alianza estratégica entre la Universidad de Los Andes, La Fundación Rafael Pombo y el Instituto Colombiano de Bienestar Familiar. Cuenta con el auspicio de COLCIENCIAS- ETI (contrato 295-97) y del ICBF- subdirección de Protección (contrato 472-97)

REFERENCIAS

i GALVIS, A.H (1.997). Micromundos lúdicos interactivos: Aspectos críticos en su diseño y desarrollo. *Informática Educativa*, **10** (2), pp. 191-204.

ii GALVIS, A.H (1992). *Ingeniería de Software Educativo.* Santafé de Bogotá: Ediciones Uniandes.

iii GAMMA, E., e*t al*. (1994). *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison Wesley.

iv GARLAN, D. , y SHAW, M. *An Introduction to Software Architectures. Technical Report CMU-CS-94-66*, School of Computer Science, Carnegie Mellon University.

v SHAW, M., GARLAR, D. (1996). *Software Architecture: Pespectives on an emerging discipline* Prentice Hall (pp. 19-32)

vi JACOBSON, I. *et. al*. (1992). *Object-Oriented Software Engineering, a LUse Case driven approach*. ACM Press, Addison-Wesley Publishing Company.

vii SHLAER, S.; MELLOR, S. (1988). *Object-Oriented Systems Analysis, Modeling the World in Data*. Yourdon Press Computing Series.

viii MARTIN, J. (1993). *Principles of Object-Oriented Analysis and Design*. Prentice-Hall.

ix FIGUEROA, P. (1997). Especificación de Software. Uniandes-LIDIE: Proyecto Ludomática (*documento de trabajo).*

x RATIONAL Home Page. *http://www.rational.com*

xi FIGUEROA, P. (1997). Metodología de desarrollo de software Orientado por Objetos. *http://agamenon.uniandes.edu.co:8088/~pfiguero/soo/metod/*

xii MOCKUS, A. (1988). *Pedagogías, escritura e informática. Educadores e Informática: promesas, dilemas y realidades*. Bogotá: Colciencias.

xiii PAPERT, S (1993). The Children's Machine. Rethinking school in the age of the computer. *Basic Books, Harper Collins Publishers*.

xiv PATTIS, R.E. (1981). *Karel The Robot: A Gentle Introduction to the Art of Programmin*g. New York: John Wiley & Sons.