Una visión a la relación entre los subgéneros literarios y la programación orientada a objetos

Ricardo de J. Botero, Tecnológico de Antioquía, Colombia
Carlos Arturo Castro, Universidad San Buenaventura Medellín, Colombia
Edgar Serna, Instituto Tecnológico Metropolitano, Colombia

Resumen: *Como el título indica, este artículo expone una visión a la relación entre una expresión humana tan fascinante y madura como la literatura, manifiesta en obras de diferentes subgéneros, y otra también atrayente de más reciente eclosión como la ciencia de la computación, manifiesta en temas relacionados con la Programación Orientada a Objetos (POO). Para establecer la relación, se presentan dos visiones generales, la primera relacionada con los subgéneros literarios y la segunda con el Lenguaje de Modelado Unificado (UML, sigla inglesa), las cuales se toman como punto de partida para establecer nexos entre las fábulas, los apartes de novela y obras literarias en general, con el modelado de software expresado en diagramas de casos de uso, clases, objetos, estados, secuencia y actividades, todos propios del UML. De ésta manera, se plantean nuevas estrategias didácticas para la comprensión del diseño orientado a objetos mediante el modelado de obras literarias.*

Palabras clave: *ingeniería de software orientada a objetos, subgéneros literarios, UML*

Abstract: *As the title suggests, this article exposes a vision to the relationship between a human expression so fascinating and mature as literature, evident in works of different genres, and other attractant also more recent hatching as computer science, manifested in themes related with the object-oriented programming (OOP). To establish the relationship, presented two overviews, the first related to the literary genres and the second with the Unified Modeling Language (UML), which are taken as a starting point to establish linkages among the fables, the asides of novel and literary works in general, with modeling of software expressed in use case diagrams, classes, objects, states, sequence and activities, all of the UML. In this way, arise new teaching strategies for the understanding of design-oriented objects through modeling of literary works.*

Keywords: *Object-Oriented Software Engineering, Literary Subgenres, UML*

Introducción

La docencia de la ingeniería de software en instituciones de educación superior requiere de didácticas innovadoras que motiven y faciliten el aprendizaje de dicho tópico. Así, los comités curriculares de programas académicos en ingeniería relacionados con ciencias de la computación, han realizado propuestas académicas para la enseñanza de asignaturas de ingeniería en software, donde estrategias formales como el Aprendizaje Basado en Problemas (Cheong, 2008), el Aprendizaje Basado en Proyectos (Breiter *et al*., 2005) (Labra *et al*., 2006) y el Aprendizaje Basado en Casos (Villalobos & Casallas, 2006) han predominado, con la incursión de otras estrategias no formales de aprendizaje enfocadas a los juegos serios (Guerrero *et al*., 2009). Ahora, se pretende exponer otra estrategia no formal ni difundida hasta el momento, que busca motivar el aprendizaje de la ingeniería de software orientada a objetos por medio de las expresiones literarias de corta extensión como la fábula y la poesía, o los fragmentos de cuento, ensayo o novela.

Establecer vínculos entre dos áreas de conocimiento disímiles entre sí puede parecer en principio extraño, dado que se trata de relacionar una de las siete artes clásicas, la literatura, con una ciencia aplicada, la ingeniería de software. El puente que facilita la relación es el Unified

Modeling Language (UML), cuya versión inicial se ofreció para estandarización al Object Management Group (OMG) por Rumbaugh, Jacobson y Booch (Rumbaugh et al, 1999), en enero de 1997; la evolución del UML ha impactado de tal forma la industria del software, que se ha convertido en el lenguaje estándar de modelado para las compañías desarrolladoras de software y en la herramienta CASE (Computer Aided Software Engineering) ad hoc de analistas y arquitectos, con versiones propietarias como Rational Rose y Microsoft Visio y de libre distribución como Argo UML y StarUML.

Dicha relación se plantea de manera tácita en un libro sobre fundamentos de programación orientada a objetos publicado por el Tecnológico de Antioquia (Botero et al, 2009), donde, a manera de motivación, cada capítulo se precede por un cuento o fragmento de obra literaria modelado con un diagrama de clases.

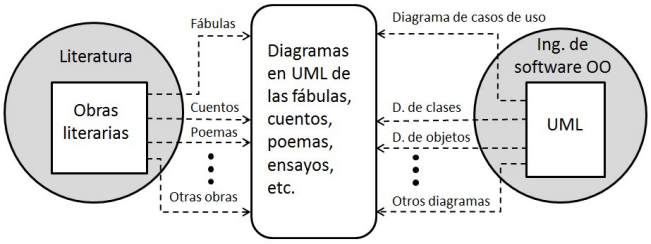

Figura 1. Relación entre la literatura y la ingeniería de software
Fuente: Botero et al., 2013.

En lo que sigue de este artículo, se presentan las visiones generales de los géneros literarios y del UML y se desarrolla la idea central que describe el nexo entre ambas temáticas, tomando como ejemplos una fábula, dos fragmentos de novela y una frase célebre. Finalmente, se presentan las conclusiones y el trabajo futuro.

Visión general de los géneros literarios

Los géneros literarios se pueden considerar "como la representación concreta del discurso humano, es decir, un acto de palabra consustancial a la capacidad del lenguaje humano" (Cerezo, 1995); constituyen grupos o categorías donde se pueden clasificar las obras literarias según su contenido y criterios semánticos, sintácticos, fonológicos, discursivos y contextuales, entre otros.

La concepción inicial de género literario se debe a Aristóteles, quien los resume en tres: épica, lírica y dramaturgia (Hernández, 2011), en su orden equivalentes a una clasificación sinónima: narrativa, poesía y teatro. El género épico relata sucesos reales o imaginarios de carácter objetivo, con algunos niveles de subjetividad; el género lírico expresa el mundo subjetivo del autor, sus sentimientos y emociones; el género dramático conlleva diálogos destinados a la representación actoral, donde se plantean diversos conflictos de la naturaleza humana.

Teniendo en cuenta aspectos sociológicos y cronológicos que no ameritan sustentación en éste artículo, es indiscutible el carácter evolutivo de las formas literarias para conformar nuevos subgéneros como la crítica, la biografía, el ensayo, la historia y la didáctica, los cuales se pueden incluir en el género épico planteado por Aristóteles. El mapa conceptual de la fig. 2 presenta los géneros y subgéneros literarios, donde los nuevos se representan en rectángulos; ésta figura excluye la oratoria por su carácter verbal, no textual.

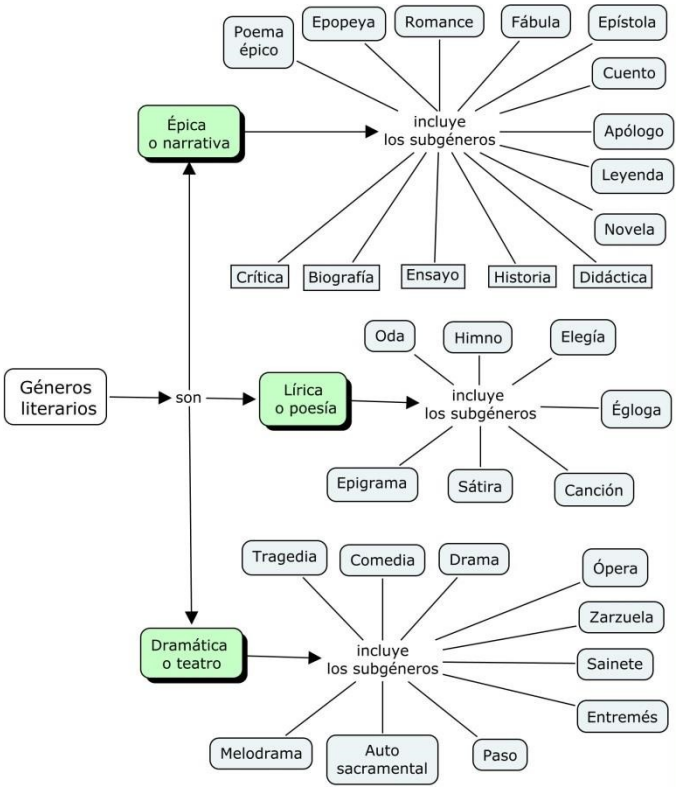

Figura 2. Géneros y subgéneros literarios
Fuente: Botero et al., 2013.

Visión general del UML

El UML es un lenguaje gráfico utilizado para modelar, especificar, visualizar, construir y documentar artefactos con gran cantidad de software; en breves palabras, es un lenguaje para construir planos de software. Los artefactos a documentar incluyen actores (personas, máquinas, software o cualquier otra entidad que pueda ejecutar una acción), clases (seres vivos como humanos, animales y plantas; elementos concretos como edificios, productos y vehículos; elementos abstractos como viajes y transacciones bancarias; o cualquier otra entidad con atributos y comportamientos propios) y objetos (elementos o ejemplares de una clase). La expresividad de éste lenguaje visual se debe a su simbología, resumida en la fig. 3, y a su modelo conceptual expuesto en el mapa de la fig. 4, el cual incluye bloques de construcción, reglas semánticas y mecanismos comunes.

Las reglas semánticas permiten diseñar modelos con bloques de construcción combinados de manera adecuada para lograr un modelo bien formado, auto consistente a nivel semántico. Los bloques de construcción incluyen relaciones entre clases, elementos estructurales, de agrupación, de comportamiento y de anotación, además de diferentes tipos de diagramas acoplables a las distintas etapas del ciclo de desarrollo de software en proyectos de mediana y gran envergadura: casos de uso, clases, objetos, estados, colaboración, secuencia, actividades, paquetes y despliegue.

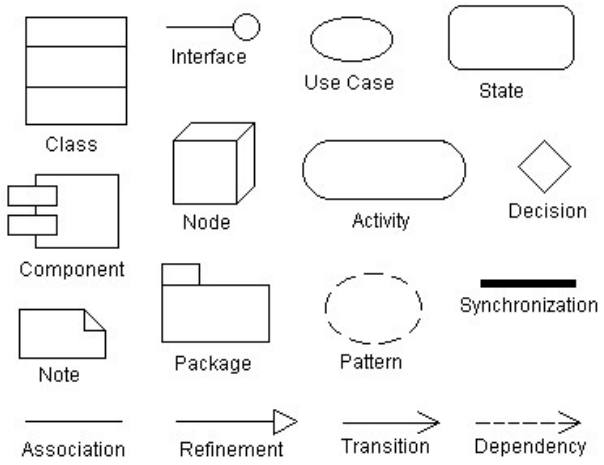

Figura 3. Principales símbolos del UML
Fuente: cs.lmu.edu, 2012.

Las reglas semánticas permiten diseñar modelos con bloques de construcción combinados de manera adecuada para lograr un modelo bien formado, auto consistente a nivel semántico.

Los mecanismos comunes le aportan más expresividad a los diferentes diagramas; consisten por lo general de palabras que aumentan la semántica del diagrama.

La idea central: relación entre los subgéneros literarios y la POO

Un título más general y conciso para el presente artículo podría ser "Una visión a la relación entre la literatura y la ingeniería de software"; sin embargo, se optó por el uso de términos más específicos como "subgéneros literarios" y "programación orientada a objetos" dado que los términos "literatura" e "ingeniería de software" conllevan una extensa definición por el amplio contexto que soportan. Así, la literatura puede comprender producciones literarias de una nación, de una época o de un género (literatura romana, literatura del siglo XV, literatura épica), y puede referirse también al conjunto de obras que versan sobre un arte o una ciencia (literatura ingenieril, literatura médica, etc.). La ingeniería de software requiere ejecutar una serie de etapas reconocidas como "ciclo de vida" enmarcadas dentro de un modelo, que incluye el análisis y especificación de requisitos, la definición de una arquitectura del sistema, la programación, las pruebas, la documentación y el mantenimiento (Sommerville, 2011).

Por tanto, los subgéneros literarios y la POO expresada con UML, constituyen los temas específicos de la literatura y la ingeniería de software, respectivamente, seleccionados para establecer la relación entre las mismas. Para ilustrar el vínculo, se tomará entre los primeros una fábula, dos fragmentos de novela y una frase célebre, por tratarse de textos breves; otros subgéneros se podrían considerar, pero son tantos que la extensión de un artículo científico es insuficiente para tratarlos. Entre la segunda se tomarán los diagramas de casos de uso, clases, objetos, estado, secuencia y actividades.

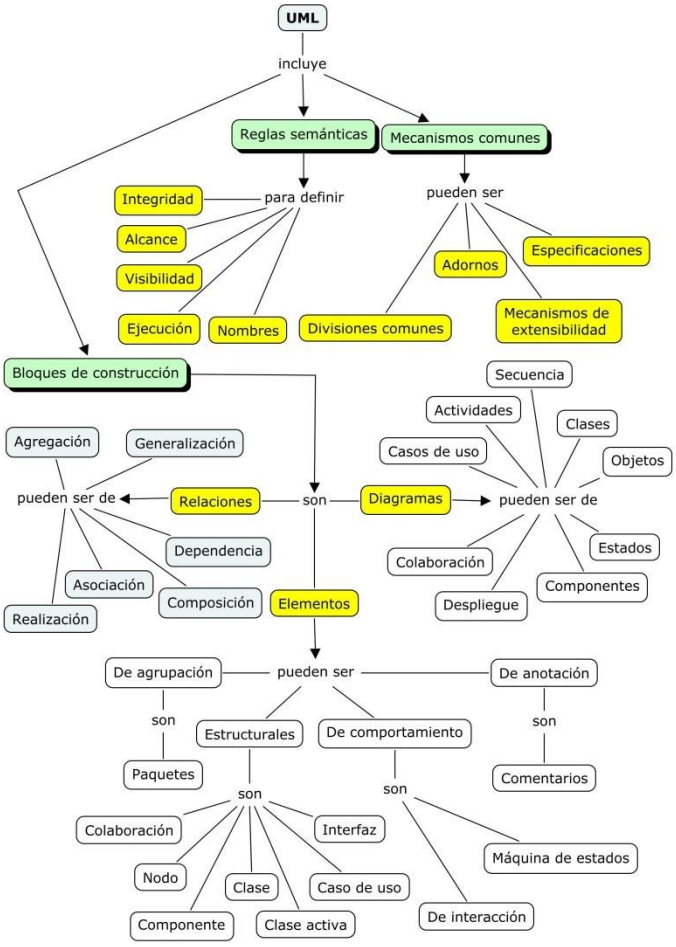

Figura 4. Modelo conceptual del UML
Fuente: Botero et al., 2013.

La fábula

El siguiente texto y moraleja corresponde a la fábula de Esopo titulada *El delfín y el mono* (Santander, 2001):

> Hay la costumbre, viajando por mar, de llevar consigo perritos de Malta y monos para distraerse durante la travesía. Un hombre que navegaba llevaba con él un mono. Al llegar a Sunión, promontorio del Ática, se desató una violenta borrasca. Se hundió el navío, y todo el mundo se salvó a nado, el mono como los demás.
>
> Vio un delfín al mono y, tomándole por un hombre, se deslizó bajo él y sosteniéndole le llevó a tierra firme. Según llegaban al Pireo, puerto de Atenas, preguntó al mono si era ateniense. Respondió el mono que sí lo era y que incluso tenía en Atenas parientes ilustres; le preguntó el delfín si también conocía el Pireo, y el mono, creyendo que le preguntaba por un hombre, le dijo que sí y que era incluso uno de sus más íntimos amigos. Indignado por tal mentira, cogió el delfín al mono, y, arrojándole al agua, le ahogó.

Se refiere esta fábula a los hombres que, sin conocer la verdad, creen poder engañar a los otros.

Figura 5. "El delfín y el mono" de Esopo[1]
Fuente: Roldán, 2013.

De manera sencilla, se pueden crear varios diagramas UML para representar la fábula desde diferentes perspectivas: estática, para identificar los objetos o entidades que intervienen (personajes por ejemplo), o dinámica, para representar las relaciones entre las entidades (conversación entre el delfín y el mono) o los estados de un objeto. Estos diagramas se dibujan primero sobre un papel y luego se pueden transcribir con una herramienta CASE (Computer Aided Software Engineering) si se prefiere[2], estableciendo de ésta forma una estrategia didáctica para la enseñanza de la ingeniería de software orientada a objetos con la inclusión de un elemento lúdico interesante: la obra literaria. La incursión de ésta lúdica en la estructura curricular de programas académicos como Tecnología en Sistemas o Ingeniería de Sistemas y Computación, por ejemplo, podría llevarse a cabo en sesiones de clase presencial o virtual, en talleres extra clase o en semilleros de investigación de ingeniería de software, con el objetivo de facilitar la comprensión de conceptos relacionados con el modelado de sistemas a partir de obras literarias. Un factor adicional que coadyuva a la formación integral del estudiante de ingeniería es la motivación por la lectura de obras literarias, efecto del vínculo que se establece entre las humanidades y la ingeniería de software.

1) Diagrama de casos de uso

El diagrama de casos de uso representa la forma como los actores interactúan con los elementos de un sistema. En la citada fábula de Esopo se identifican cuatro actores: hombre, mono, delfín y perrito de Malta, aunque este último es intrascendente en la narración, tal cual se refleja en la fig. 6, donde no aparece; esto es análogo al proceso de abstracción en ingeniería de software: se eligen las clases importantes y las superfluas se descartan.

Cada actor ejecuta acciones que se representan en óvalos, denominados casos de uso. Nótese que el caso de uso *Conversar* incluye otros dos, *Preguntar al mono* y *Responder al delfín*; además, el caso de uso *Mentir* es una extensión de *Responder al delfín*.

2) Diagrama de clases

Un diagrama de clases es un modelo de vista estático de un sistema, que incluye clases representadas por rectángulos relacionados entre sí. Una clase es la descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica.

[1] Las ilustraciones de las figuras 5, 13 y 18, fueron realizadas por Jaime Roldán Arias, profesor universitario de matemáticas en el Tecnológico de Antioquia.
[2] Los diferentes diagramas de éste artículo se graficaron con Argo ULM v0.34 y Visual Paradigm for UML Community Edition v10.1.

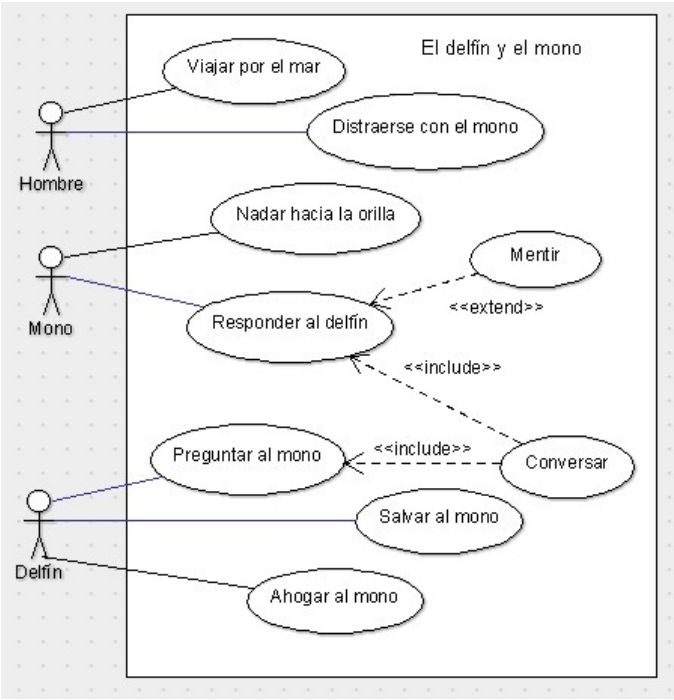

Figura 6. Diagrama de casos de uso para la fábula "El delfín y el mono"
Fuente: Botero et al., 2013.

La fig. 7 presenta el diagrama de clases para la fábula *El delfín y el mono*, donde se observan relaciones de generalización o herencia, agregación, asociación y dependencia. Las clases *PerritoDeMalta*, *Mono* y *Delfín* heredan de la clase abstracta *Animal* (el símbolo del UML para la herencia es una flecha con punta triangular); las clases *Puerto* y *Promontorio* forman parte de la clase abstracta *Tierra* (la relación de agregación se representa con una línea que inicia en la "parte" y finaliza en el "todo" con una punta en forma de rombo); el atributo *dueño* de las clases *PerritoDeMalta* y *Mono* forma dos relaciones de asociación entre estas clases y la clase *Hombre* (flechas con línea sólida); en último lugar, se muestran varias relaciones de dependencia (representadas por flechas con línea segmentada) etiquetadas con los verbos conversar, viajar, desembarcar y chocar.

Cabe aclarar que en el diagrama de clases de la fig. 7, la clase *PerritoDeMalta* se puede omitir teniendo en cuenta que no se tomó como actor en el diagrama de casos de uso expuesto en la fig. 6 y en aras de conservar la correspondencia o acoplamiento entre diagramas del UML en el proceso de modelado de un sistema. Sin embargo, se incluye con propósito didáctico porque, en efecto, un perrito de Malta es un animal. Lo mismo se puede decir de las clases *Tierra, Puerto, Promontorio, Borrasca y Navío*.

3) Diagrama de estados

Los diagramas de estados se utilizan para modelar los aspectos dinámicos de los objetos de un sistema, es decir, representan los diferentes estados de un objeto específico a lo largo de su ciclo de vida. Dado que cada clase aglutina un conjunto de objetos, se podrían plantear para el caso de la fábula en cuestión hasta ocho diagramas de estados, según se observa en el diagrama de

clases de la fig. 7; se excluyen las clases abstractas *Animal* y *Tierra* porque a partir éstas —por su calidad de abstractas— no se pueden instanciar objetos.

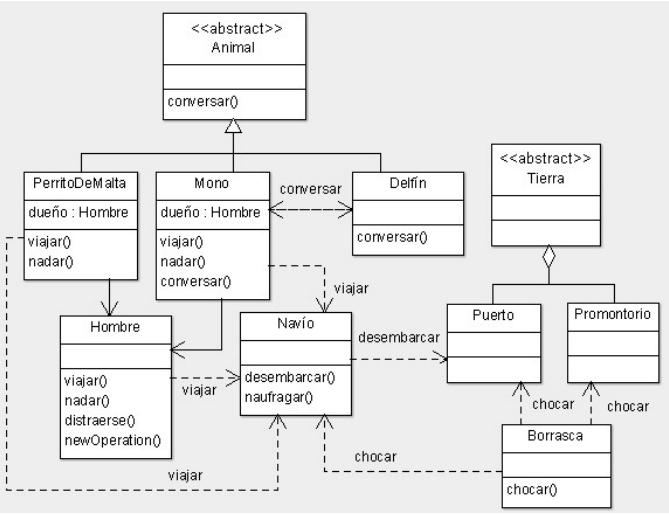

Figura 7. Diagrama de clases para la fábula "El delfín y el mono"
Fuente: Botero et al., 2013.

Todos los delfines y monos de nuestro planeta son objetos o instancias de las clases *Delfín* y *Mono*, respectivamente. En términos de software, para la fábula de Esopo se identificará al delfín como el objeto *d* y al mono como el objeto *m*, los cuales pasan por diferentes estados según discurre la fábula; dichos estados, simbolizados por rectángulos con vértices redondeados como se muestra en las figuras 8 y 9, representan los estados del objeto *d* o *m* en respuesta a los sucesos y al tiempo; el círculo relleno indica el inicio de la secuencia de estados y una diana representa su finalización. La transición de un estado a otro se simboliza con una flecha desde el estado anterior al siguiente; dos estados unidos por una flecha se denominan secuenciales, por tanto un diagrama de estados se puede considerar como un conjunto de estados secuenciales.

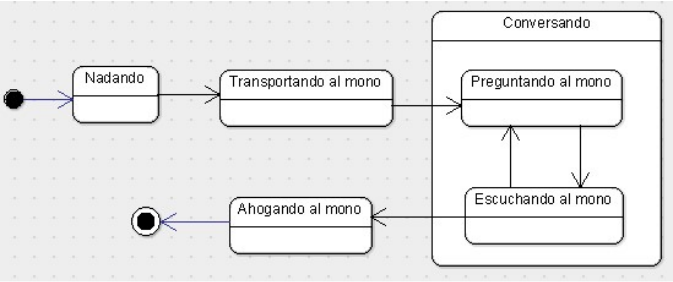

Figura 8. Diagrama de estados para el objeto d (instancia de Delfín)
Fuente: Botero et al., 2013.

Un micro o sub-estado es un estado al interior de otro y un macro estado es aquel que contiene a otros. Para el objeto *d* se presenta el macro estado *Conversando*, con dos micro estados secuenciales: *Preguntando al mono* y *Escuchando al mono*; el objeto *m* también pasa por un macro estado con el mismo nombre, *Conversando*, pero con micro estados secuenciales complementarios: *Escuchando al delfín* y *Respondiendo al delfín*.

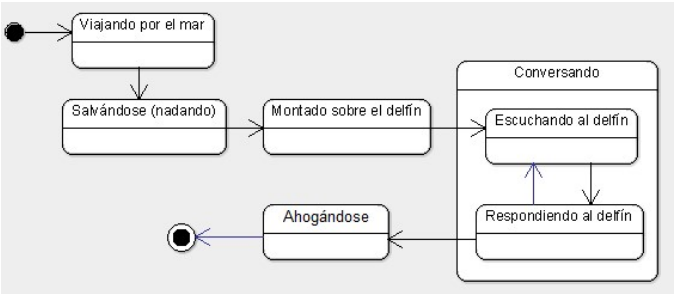

Figura 9. Diagrama de estados para el objeto m (instancia de Mono)
Fuente: Botero et al., 2013.

4) Diagrama de secuencias

Los diagramas de estado representan los diferentes estados de un objeto. Un diagrama de secuencias establece el paso consecutivo: ilustra la forma en que los objetos se comunican entre sí al transcurso del tiempo. La fig. 10 ilustra como una borrasca choca contra el navío, suscitando una alarma —dígase pánico— en el mono, que lo incita a nadar; luego, aparece el delfín que lo salva, viene el acto de conversación entre ambos animales y finalmente el ahogamiento del mono a iniciativa del delfín.

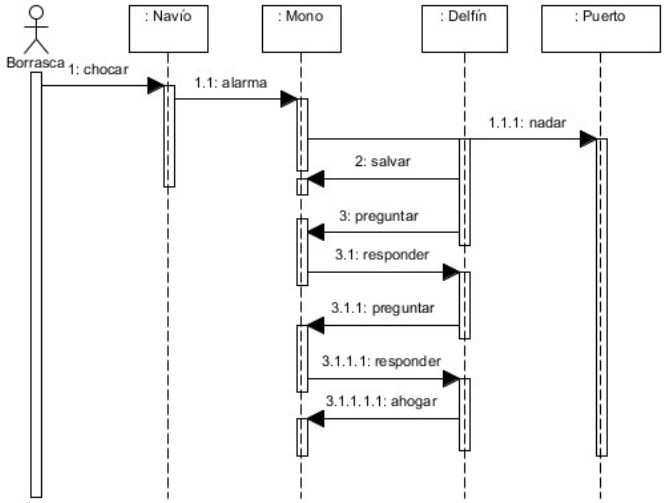

Figura 10. Diagrama de secuencia para la fábula "El delfín y en mono"
Fuente: Botero et al., 2013.

5) Diagrama de actividades

Un diagrama de actividades representa el comportamiento interno de una operación o de un caso de uso, que muestra el flujo de control entre actividades de manera secuencial.

La fig. 11 expone un diagrama de actividades para el caso de uso *Salvar al mono* del objeto *d* (delfín), donde el inicio, el fin del diagrama y la transición de actividad se representan de manera similar que en el diagrama de estados –un círculo relleno, una diana y una flecha, res-

pectivamente–. Cada actividad se representa con un óvalo y la bifurcación o toma de decisiones con un rombo.

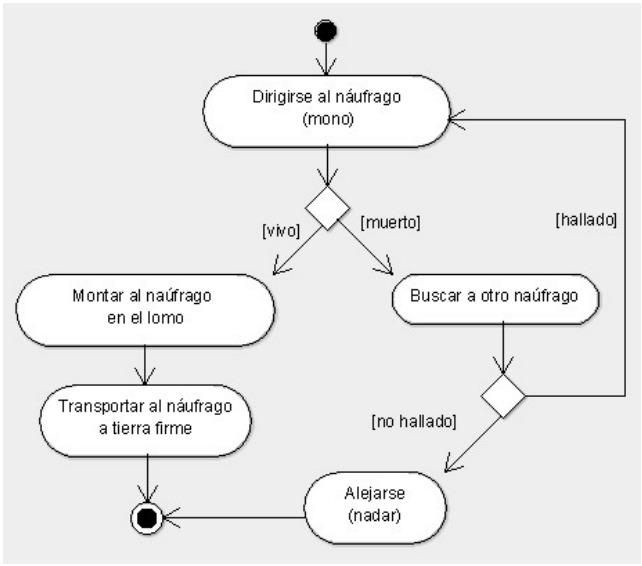

Figura 11. Diagrama de actividades para el caso de uso "Salvar al mono" del objeto d (delfín).
Fuente: Botero et al., 2013.

Una descripción en lenguaje natural del diagrama de actividades de la fig. 11 es la siguiente: el delfín se dirige al náufrago —en éste caso el mono—, y verifica si está vivo para proceder a salvarlo; si el náufrago está muerto, el delfín pasa a buscar otro náufrago; al encontrarlo, se repiten las actividades de salvamento. Si no se encuentra un náufrago, el delfín sigue en su actividad cetácea rutinaria.

En la fig. 12 se observa el diagrama de actividades para el caso de uso *Responder al delfín*, que presenta sólo dos estados y una bifurcación.

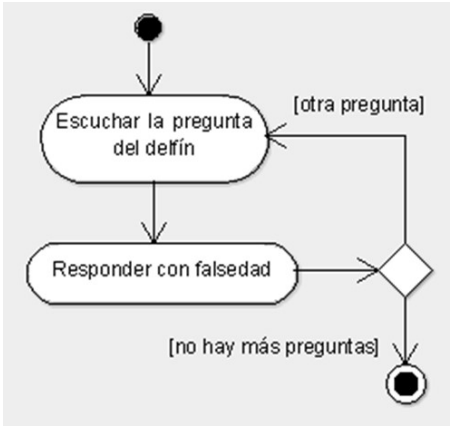

Figura 12. Diagrama de actividades para el caso de uso "Responder al delfín"
Fuente: Botero et al., 2013.

La novela

Representar en UML los diferentes actores, actividades y relaciones entre los personajes de una novela, puede convertirse en un trabajo oneroso para el analista debido a la extensión de las obras de éste subgénero literario. ¡Imagínese un diagrama de clases para Los Miserables de Víctor Hugo!

La sugerencia para el caso de la novela, los cuentos largos o cualquier otra obra literaria extensa, es tomar pasajes o apartados que tengan sentido por sí mismos, que se presten para la comprensión de algunos conceptos de la POO a través del modelado con el UML, y que ofrezcan posibilidades didácticas para el profesor y en consecuencia de comprensión y motivación para los estudiantes.

Para ilustrar lo anterior tomaremos dos pasajes, en éste caso conversaciones de *A través del espejo y lo que Alicia encontró allí*, (Carroll, 1984), y de *El Ingenioso Hidalgo Don Quijote de la Mancha* (De Cervantes, 1980).

a) Alicia

El siguiente es un fragmento de la novela *A través del espejo y lo que Alicia encontró allí*:

> Ahora está soñando —señaló Tarará— ¿y a que no sabes lo que está soñando?
> — ¡Vaya uno a saber! —replicó Alicia— ¡Eso no podría adivinarlo nadie!
> — ¡Anda! ¡Pues si te está soñando a ti! —exclamó Tarará batiendo palmas en aplauso de su triunfo—. Y si dejara de soñar contigo, ¿qué crees que te pasaría?
> — Pues que seguiría aquí tan tranquila, por supuesto —respondió Alicia.
> — ¡Ya! ¡Eso es lo que tú quisieras! —replicó Tarará con gran suficiencia—. ¡No estarías en ninguna parte! ¡Como que tú no eres más que un algo con lo que está soñando!
> — Si este Rey aquí se nos despertara —añadió Tararí— tú te apagarías… ¡zas! ¡Como una vela!".

Figura 13. Conversación entre Alicia, Tararí y Tarará
Fuente: Roldán, 2008.

De igual manera que en la fábula de Esopo, podríamos realizar varios diagramas en UML para representar aspectos estáticos y dinámicos de este fragmento de conversación entre Alicia,

Tararí y Tarará[3]. Sin embargo, sólo se presentarán los diagramas de casos de uso y estado para los diversos personajes.

La fig. 14 expone los casos de uso para Alicia, el Rey, Tararí y Tarará, donde además de los actores y casos de uso se observan estereotipos <<*extend*>>: los casos de uso *Afirmar*, *Exclamar* y *Preguntar* son extensiones del caso de uso *Conversar*. Observar el comentario que aparece en la esquina superior derecha de la figura, rectángulo que asemeja un papel con una esquina doblada, utilizado para realizar cualquier tipo de aclaración al analista. Todo diagrama del UML admite comentarios.

Cuando un diagrama se hace extenso o complejo por la cantidad de relaciones entre sus elementos, puede resultar conveniente dividirlo. Por ejemplo, aunque el diagrama de la fig. 14 no es extenso ni complejo, los casos de uso se pueden separar por actor como se observa en la fig. 15, aportándoles mayor claridad (se omite al actor Rey porque es independiente desde la fig. 14).

La fig. 16 presenta los diagramas de estado para dos objetos (el Rey y Alicia), donde es importante resaltar la transición recursiva para el estado *Durmiendo* (transición representada por una flecha que parte y llega al mismo estado) y la restricción etiquetada por la disyunción *or* encerrada entre llaves, para dar a entender que Alicia puede pasar al estado de *Apagándose* (desapareciendo) sea cuando está *Escuchando* o cuando está *Respondiendo*.

La fig. 17 muestra los diagramas de estado para los otros dos objetos de la conversación: Tararí y Tarará. Los estados de Tararí son secuenciales y de simple interpretación; sin embargo, para el objeto Tarará se presentan dos estados concurrentes, que ocurren al mismo tiempo, dado que Tarará exclama y bate las palmas a la vez; los estados concurrentes se delimitan por un par de barras gruesas y paralelas.

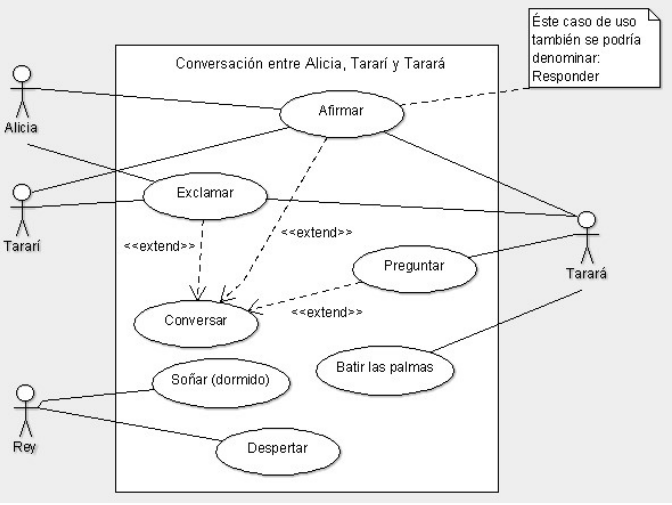

Figura 14. Diagrama de casos de uso para la conversación entre Alicia, Tararí y Tarará
Fuente: Botero et al., 2013.

[3] Los nombres originales en inglés de Tararí y Tarará son Tweedledum y Tweedledee; otras traducciones castellanas apuntan a denominaciones como Patachunta y Patachún o Domisol y Solmidó.

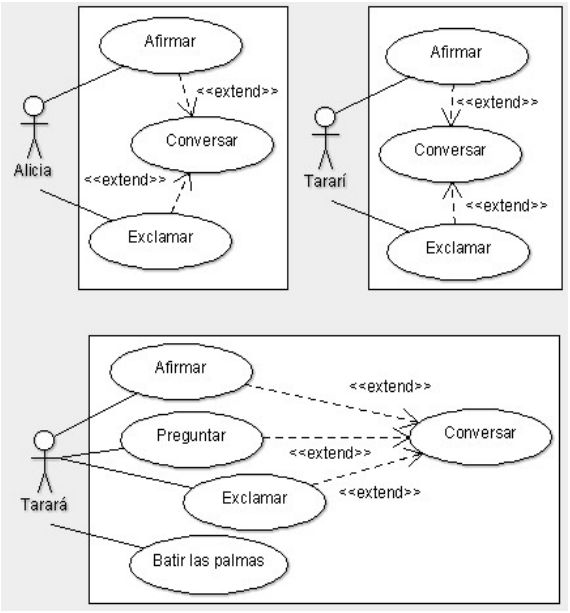

Figura 15. Diagramas de casos de uso por objeto, para la conversación entre Alicia, Tararí y Tarará
Fuente: Botero et al., 2013.

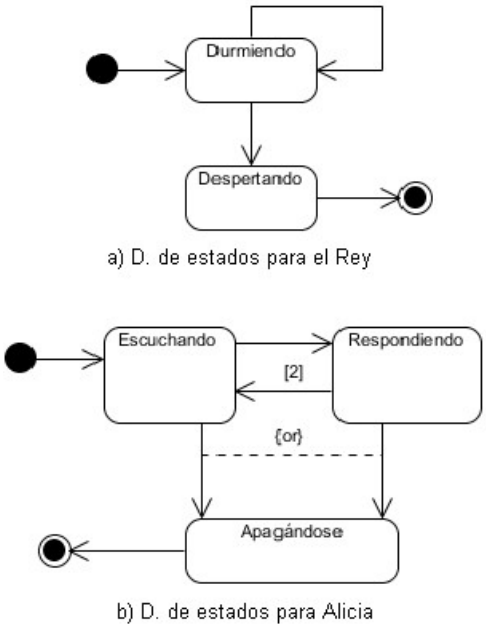

Figura 16. Diagramas de estados para el a) el Rey y b) Alicia
Fuente: Botero et al., 2013.

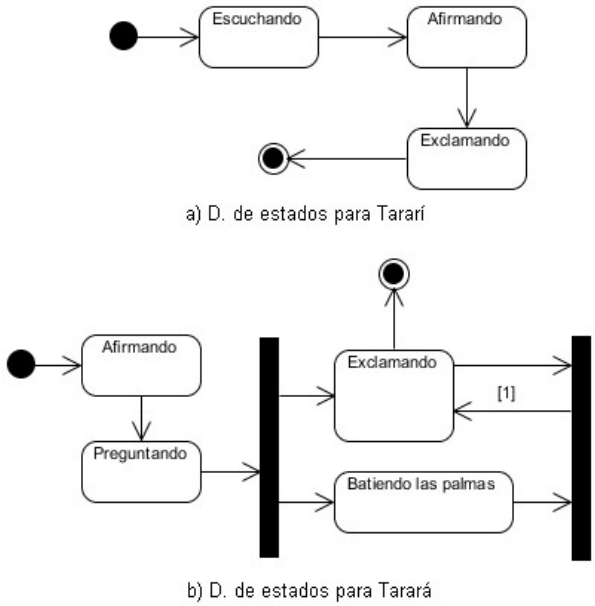

Figura 17. Diagramas de estados para Tararí y Tarará
Fuente: Botero et al., 2013.

b) Don Quijote

El siguiente es un fragmento del Capítulo XXXV de la novela El Ingenioso Hidalgo Don Quijote de la Mancha (*Donde se prosigue la noticia que tuvo Don Quijote del desencanto de Dulcinea, con otros admirables sucesos*):

> Tentóse, oyendo esto, la garganta Don Quijote, y dijo, volviéndose al duque:
> — Por Dios, señor, que Dulcinea ha dicho la verdad: que aquí tengo el alma atravesada en la garganta, como una nuez de ballesta.
> — ¿Qué decís a esto, Sancho? –preguntó la duquesa.
> — Digo, señora –respondió Sancho–, lo que tengo dicho; que de los azotes, abernuncio.
> — «Abrenuncio» habréis de decir, Sancho, y no como decís –dijo el duque.

Figura 18. Conversación entre Don Quijote, la duquesa, Sancho y el duque
Fuente: Roldán, 2013.

En la fig. 19 se muestra el diagrama de clases para este fragmento de novela, donde se presentan clases abstractas y relaciones de herencia y dependencia. Para el caso concreto de la clase *Hidalgo* se observa que hereda de *MiembroNobleza*, que a su vez hereda de *Persona*; además, la clase *Hidalgo* constituye el eje central del diagrama, en razón que el hidalgo es nombrado caballero, tiene un escudero a su servicio, conversa con los duques y está enamorado de Dulcinea. Cabe aclarar dos asuntos: primero, que ésta jerarquía de clases no se puede abstraer de la simple lectura del fragmento, sino que requiere de una lectura completa del capítulo o del conocimiento previo del mundo de su protagonista, Alonso Quijano, don Quijote; y segundo, dentro del proceso de abstracción, las clases *Garganta, NuezDeBallesta y Azote*, son irrelevantes.

Teniendo en cuenta que todo diagrama de clases conlleva uno o varios diagramas de objetos y que una clase está formada por un conjunto de objetos que comparten las mismas características y operaciones, se presenta el diagrama de objetos de la fig. 20, donde el orden de los mensajes representados por flechas se encuentra numerado según el desenlace de la conversación. Notar que los objetos *elDuque* y *laDuquesa* pertenecen a la misma clase: *Duque*.

Frase célebre

Los diagramas de objetos ofrecen particular interés. Por ejemplo, la frase célebre de Oscar Wilde "Amarse a sí mismo es el comienzo de una aventura que dura toda la vida" (Jordà & Català, 2009), conlleva al diagrama de clases de la fig. 21, donde se observa una asociación recursiva, y al de objetos de la fig. 22, donde se exponen once objetos, tres de tipo *Persona* ocho de tipo *Actividad*.

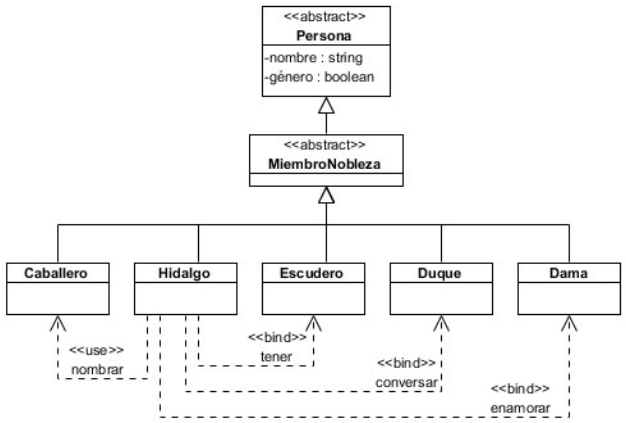

Figura 19. Diagrama de clases para un fragmento de Don Quijote
Fuente: Botero et al., 2013.

Una asociación recursiva apunta a una misma clase y se puede presentar en otro tipo de diagramas como el de estados, expuesto en la fig. 16. Para el caso de la fig. 21, la recursión se presenta porque para una persona en particular, su madre y padre también son personas; la multiplicidad de la asociación se presenta de cero a dos (0..2) para evitar una recursión infinita, es decir, se puede presentar en el contexto del software el caso hipotético de una persona sin padre y madre, que en términos biológicos es inadmisible.

La multiplicidad de la relación de dependencia entre las clases *Persona y Actividad* es de uno a muchos (1..*), porque una persona puede desempeñar una o más actividades, como se observa en la fig. 22 donde el objeto *x*, llamado Juan, desempeña varias actividades: meditar, estudiar, jugar, trabajar, practicar deporte y entretenerse.

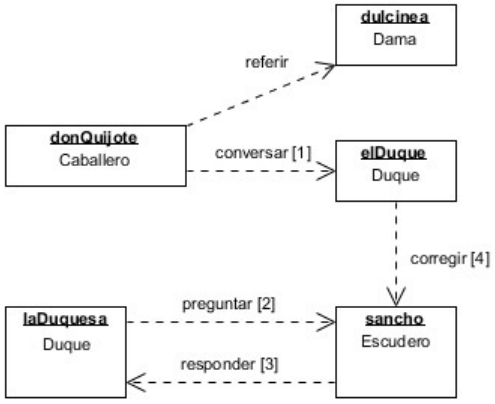

Figura 20. Diagrama de objetos para un fragmento de Don Quijote
Fuente: Botero et al., 2013.

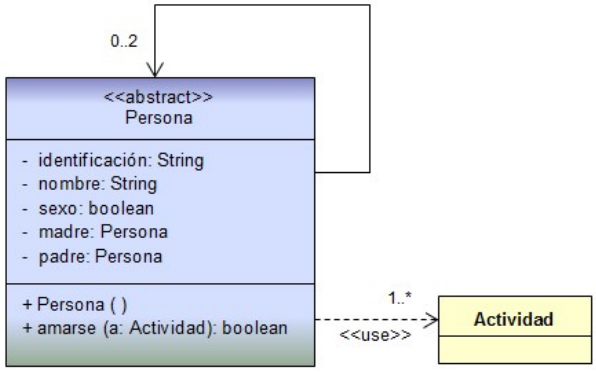

Figura 21. Diagrama de clases para una frase célebre
Fuente: Botero et al., 2013.

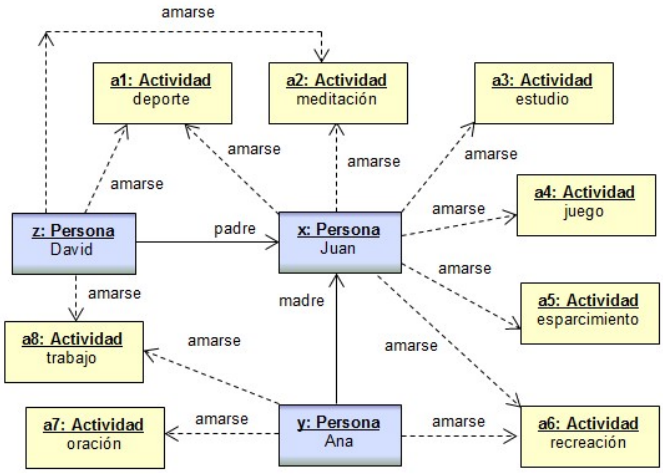

Figura 22. Diagrama de objetos para una frase célebre
Fuente: Botero et al., 2013.

Otros subgéneros literarios

Así como se han dado ejemplos de la aplicación del UML con una fábula, dos fragmentos de novela y una frase célebre, de igual manera se podrían plantear ejemplos con obras pertenecientes a otros subgéneros literarios como la epístola, la leyenda, la biografía, la tragedia, el sainete, el himno, la égloga o cualquier otro subgénero. Lo importante es elegir el texto apropiado, de tal modo que se facilite el proceso de abstracción de clases, objetos, estados y demás elementos de modelado, permitiendo una comprensión de los diferentes diagramas del UML y de la programación orientada a objetos mediante obras literarias.

Conclusiones

Las conclusiones se pueden plantear desde dos perspectivas: la del ingeniero de software y la del escritor humanista. En ambientes educativos, la primera perspectiva incluye estudiantes y profesores de tecnología en sistemas, ingeniería informática, ingeniería en software y áreas afines; la segunda incluye estudiantes y profesores de literatura, lenguas nativas y humanidades en general.

En relación a los estudiantes de ingeniería en software y programas académicos relacionados, la comprensión de la POO y del UML a partir de obras literarias o fragmentos de ellas, se puede facilitar y motivar en diferentes espacios académicos como las sesiones de clase, los talleres para resolver en el tiempo de trabajo independiente o en las actividades propias de los semilleros de investigación asociados con ingeniería de software. El nexo entre la literatura y la programación orientada a objetos, motiva la lectura de obras literarias en estudiantes y profesores de ingeniería en software, humanizando de ésta manera un área per se técnica. Además, si se utilizan ilustraciones de la obra literaria, se está generando un apoyo subrepticio hacia la obra pictórica.

En relación con los estudiantes y profesores de humanidades, se puede plantear que a partir del nexo Obra literaria-UML, se facilita la comprensión de un lenguaje de modelado utilizado a nivel científico para documentar software, acercándolos de ésta manera al mundo tecnológico relacionado con las ciencias computacionales.

Trabajo futuro

Para el caso del TdeA, USBMed e ITM, instituciones donde se encuentran vinculados los profesores autores del presente artículo, se llevarán a cabo actividades iniciales con los semilleros de investigación relacionados con ingeniería de software, donde se presenten obras literarias cortas de diversa índole, a partir de las cuales se explicarán los conceptos fundamentales de la ingeniería de software orientada a objetos con UML. Estas actividades dinamizarán los semilleros de investigación, porque las obras literarias instauran elementos lúdicos que coadyuvan a la comprensión del paradigma de programación orientado a objetos.

En el currículo de asignaturas relacionadas con ingeniería de software se pueden estudiar casos, problemas y ejemplos convencionales, además de otros que encaminen a las obras literarias, con el objetivo de plantear talleres de ejercicios propuestos a resolver por los estudiantes en su tiempo de trabajo independiente.

También en otros congresos y eventos científicos relacionados con educación en ingeniería de software o en humanidades, se pueden presentar ponencias donde se modelen con el UML obras literarias distintas a las expuestas en el presente artículo, escritas por autores selectos o anónimos.

REFERENCIAS

Botero, R., Castro, C., Taborda, G., Maya, J., Valencia, M. (2009). *Lógica y programación orientada a objetos: un enfoque basado en problemas.* Grupo GIISTA. Tecnológico de Antioquia Institución Universitaria, Medellín.

Breiter, A., Fey, G., Drechsler, R. (2005). "Project-Based Learning in Student Teams in Computer Science Education." *Facta Universitatis – Series: Electronics and Energetis* 18(2): 165-180.

Carroll, L. (1984). *A través del espejo y lo que Alicia encontró allí*. Madrid: Ediciones Generales Anaya.

Cerezo, M. (1995). "Aristóteles y la teoría del género literario". *Revista Faventia* 17(2): 33-34.

Cheong, F. (2008). "Using a Problem-Based Learning Approach to Teach an Intelligent Systems Course." *Journal of Information Technology Education* 7.

De Cervantes, M. (1980). *El Ingenioso Hidalgo Don Quijote de la Mancha*. Espasa - Calpe / Afanias, Madrid.

Guerrero, D., Trefftz, H., Anaya, R. (2009). "Juegos en la enseñanza de la ingeniería de software". *Revista Tecno Lógicas*, n° 22, Instituto Tecnológico Metropolitano.

Hernández, A. (2011). "Breviario sobre la teoría de los géneros literarios". *Revista Luthor* 1(4).

Jordà, V., Català, X. (2009). Portal Proverbia. Autores/Wilde, Oscar. Novixar. En línea: http://www.proverbia.net/citasautor.asp?autor=1058

Labra, J. E. *et al.* (2006). "Una Experiencia de aprendizaje basado en proyectos utilizando herramientas colaborativas de desarrollo de software libre". XII Jornadas de Enseñanza universitaria de la Informática, JENUI 2006.

Loyola Marymount University. *An overview of UML.* Recuperado de: http://cs.lmu.edu/~ray/notes/umloverview/ Fecha consulta: 2 de mayo de 2013.

Rumbaugh, J., Jacobson, I., Booch, G. (1999). *The Unified Modeling Language Reference Manual*. 2nd edition. Addison- Wesley.

Santander, M.L. (2001). *Las Mejores Fábulas. Antología*. Chile: Pehuén Editores.

Sommerville, I. (2011). *Software Engineering*. 9th edition. Addison- Wesley.

Villalobos, J.A., Casallas, R. (2006). *Fundamentos de programación. Aprendizaje activo basado en casos.* Prentice Hall.

SOBRE LOS AUTORES

Ricardo de J. Botero: Ingeniero de Sistemas, Especialista en Didáctica Universitaria y Magíster en Ingeniería (área Sistemas y Computación). Profesor vinculado a la Facultad de Ingeniería del Tecnológico de Antioquia en Medellín, Colombia. Se ha desempeñado por 18 años como catedrático en varias universidades del área metropolitana de Medellín. Sus áreas de docencia son la lógica de programación, la ingeniería de software, las matemáticas discretas y el análisis de algoritmos. Las áreas de interés investigativo son la programación orientada a objetos y las estructuras y bases de datos. Ha publicado tres libros y varios artículos en revistas indexadas, con presentaciones en eventos nacionales e internacionales relacionados con la ingeniería de sistemas y computación.

Carlos Arturo Castro: Ingeniero de Petróleos y Especialista en Sistemas de Información Geográfica; candidato a Magíster en Ingeniería Informática. Con siete años de experiencia en educación básica secundaria y más de 17 años como docente investigador en educación superior. Vinculado a la Universidad de San Buenaventura Medellín. Las áreas de interés incluyen fun-

damentos de programación, estructuras de datos, bases de datos, ingeniería de software, sistemas de información geográfica y la educación informática, sobre los que ha publicado varios artículos y ha participado con ponencias en eventos nacionales e internacionales.

Edgar Serna: Científico computacional teórico, Ingeniero de Sistemas y Magister en Ingeniería de Sistemas, con más de 10 años de experiencia en la industria como líder de proyectos en Sistemas de Información y Arquitecto de Software, y profesor universitario e investigador con más de 20 años de trayectoria. Sus áreas de investigación son la Lógica, la Ingeniería de Software, las Ciencias Computacionales, los Métodos Formales y las Matemáticas en la Computación, alrededor de las cuales ha publicado libros y artículos, y participado con ponencias y conferencias en eventos nacionales e internacionales.