Rev. Téc. Ing. Univ. Zulia. Vol. 31, Nº 1, 3 - 12, 2008

Consistency game: a didactic strategy for software engineering*

Carlos Mario Zapata Jaramillo y Mary Inés Duarte Herrera

Grupo de Investigación en Ingeniería de Software, Escuela de Sistemas, Universidad Nacional de Colombia, sede Medellín. Carrera 80 # 65-223 Bloque M8. Barrio Robledo, Medellín, Antioquia, Colombia. Fax: 425 53 65, Telf: 425 53 50. {cmzapata, miduarte}@unalmed.edu.co

Abstract

Software Engineering teaching has been traditionally worked with conventional methods like magisterial classes and practical projects. These strategies have been centered in teachers as learning sources, though they have permitted professional formation through four decades, but they apart student-centered learning strategies, like games, case studies, and simulations. In this paper we propose "Consistency Game", a way to achieve students learning from a playful experience; the game works with the concept UML (Unified Modeling Language) diagrams consistency. UML is, nowadays, the most accepted standard for information systems modeling in Software Engineering.

Key words: Consistency, games, UML, software engineering.

El juego de la consistencia: una estrategia didáctica para la ingeniería de software

Resumen

La enseñanza de la ingeniería de software se ha abordado tradicionalmente con métodos convencionales como clases magistrales y proyectos prácticos. Estas estrategias, si bien han permitido la formación de profesionales en esta área durante las últimas cuatro décadas, se han centrado en el docente como fuente de aprendizaje y dejan de lado estrategias en las cuales los alumnos son el centro del aprendizaje, tales como los juegos, estudios de casos y simulaciones. En este artículo se propone "El Juego de la Consistencia", como una manera de lograr el aprendizaje de los alumnos desde una experiencia lúdica; el juego trabaja el concepto de consistencia entre diagramas de UML (Unified Modeling Language), el estándar más aceptado en la actualidad para el modelado de sistemas informáticos en ingeniería de software.

Palabras clave: Consistencia, juegos, UML, ingeniería de software.

1. Introducción

La ingeniería de software es una de las áreas principales de la ingeniería, la informática y las ciencias de computación, que brinda métodos y técnicas para desarrollar y mantener software de calidad. Además, aborda todas las fases del ciclo de vida del desarrollo de cualquier tipo de sistema de información. Su aplicación se puede notar en áreas tan diversas como la banca, la investigación científica, el control de tráfico, entre otras [1]. Zelkowitz [2] y Boehm [3] han defini-

* Este artículo se realizó en el marco de los siguientes proyectos de investigación: "Construcción automática de esquemas conceptuales a partir de lenguaje natural", financiado por la DIME y "Definición de un esquema preconceptual para la obtención automática de esquemas conceptuales de UML", financiado por DINAIN y administrado por la DIME.

do la ingeniería de software como el estudio de los principios y las metodologías que se requieren, no sólo para el desarrollo, sino también para el mantenimiento de sistemas de software, incluyendo la documentación asociada con el desarrollo. Lo que se pretende al poner en práctica la ingeniería de software, es obtener programas computarizados que sean funcionales, estables, confiables y seguros [4] y para ello utiliza el modelado (un conjunto de abstracciones o realidades seleccionadas que se construyen para entender un problema) como herramienta para la solución de problemas de la vida diaria, antes de implementar una solución. En Ingeniería de Software, el modelado se ha apoyado en análisis y diseño orientado a objetos [5], y más recientemente en UML [6], cuya versión 2.0 se basa en tres clases de diagramas: estructurales, de interacción y de comportamiento. Estos diagramas se pueden usar conjuntamente para modelar el mismo problema desde diferentes ópticas o puntos de vista, que pueden de tipo estructural o dinámico dependiendo del interesado que los emplee; para lograr este objetivo, tales diagramas deben contar con consistencia, es decir con coherencia entre los mismos, de forma que los requisitos de una especificación no se contradigan entre sí [7].

Zapata y Awad [8] afirman que los ingenieros de software deben tener aptitudes de tipo administrativo, que poco se cultivan en la enseñanza tradicional. Hoy en día, la enseñanza no se basa sólo en los conceptos impartidos por el maestro, sino que se vuelca hacia el estudiante como elemento central de la clase. Es por ello que ahora se emplean los juegos como herramienta pedagógica.

En este artículo se define "El juego de la Consistencia", una herramienta didáctica, para la enseñanza de ingeniería de software. El artículo está organizado de la siguiente forma: en la Sección 2 se muestra el marco conceptual en el que se mueve el juego; en la Sección 3 se discute el papel de los juegos en la enseñanza; en la Sección 4 se presenta el diseño experimental de "El juego de la Consistencia", seguido por la Sección 5 donde se ven algunos resultados de la aplicación del juego a varios grupos de participantes; en la Sección 6 se presentan las conclusiones y, finalmente, en la Sección 7 se discute el trabajo futuro que se deriva de este juego.

2. Marco Teórico

2.1. Esquemas preconceptuales

Los esquemas preconceptuales [9] se basan en los grafos conceptuales (CGs) que fueron ideados por John F. Sowa [10]. Estos grafos sirven como un lenguaje intermedio entre la lógica formal y el lenguaje natural. Los esquemas preconceptuales emplean conceptos (rectángulos) para representar sustantivos y relaciones (óvalos) para representar verbos; ambos elementos se unen mediante flechas que representan la asociación entre ellos. Dichas flechas sólo pueden ir de relaciones a conceptos y viceversa. Estos esquemas también sirven como medio de comunicación entre los interesados (quienes tienen algún tipo de interés en desarrollar una pieza de software) y los analistas (encargados de su modelado). Un esquema preconceptual puede representar todo el discurso –expresando todas las frases resumidas en un solo grafo– incluyendo las situaciones dinámicas que se presentan en el discurso, combinando estructura y comportamiento simultáneamente.

Un ejemplo del esquema preconceptual se aprecia en la Figura 1. Se pueden ver situaciones dinámicas que se muestran en la implicación (flecha gruesa); se señalan también los principales elementos de dicho diagrama.

2.2. UML

El UML es un lenguaje gráfico de modelado que cubre varias fases en el proceso de desarrollo de una pieza de software. Este lenguaje fue adoptado como un estándar por el OMG (Object Management Group) y ayuda a especificar, visualizar y documentar modelos de sistemas de software. Se puede usar UML para modelar negocios o algunos sistemas cuya solución no necesariamente sea una pieza de software [6]. En su versión 2.0, UML define 13 diagramas, que se dividen en tres categorías: estructura (clases, objetos, componentes, estructura de composición, paquetes, despliegue), que representan los elementos de una especificación sin tomar en consideración el tiempo, comportamiento (casos de uso, máquina de estados y actividades), que describen las características dinámicas de un objeto, tales como sus operaciones y métodos, e interacción (se-

cuencias, comunicación, vista de interacción y tiempos), que se derivan de los diagramas de comportamiento y que hacen énfasis en la manera como los objetos interactúan entre sí [11]. A continuación, se detallan los diagramas que se usan en el Juego de la Consistencia.

- Diagrama de casos de uso. Ayuda al cliente, al interesado y a los desarrolladores a utilizar el sistema. Cada tipo de usuario se representa mediante un actor, quien utiliza el sistema al interactuar con los casos de uso, los cuales representan los requisitos funcionales del sistema. En la Figura 2 se muestra un ejemplo del diagrama de casos de uso.
- Diagrama de clases: Se utiliza para mostrar clases y sus relaciones. Una clase es una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica. Un atributo es una propiedad que describe el rango de valores que los objetos pueden tomar. Una operación es la implementación de un servicio que puede ser solicitado por cualquier objeto de la clase para afectar su comportamiento. En la Figura 3 se muestra un ejemplo del diagrama de clases.
- Diagrama de secuencias: Hace énfasis en la ordenación temporal de los mensajes. Muestra cómo el control se transmite de un objeto a otro a medida que se envían mensajes. Un mensaje enviado por un objeto dispara la toma del control en el objeto receptor y la realización de las operaciones de su clase. En la Figura 4 se presenta ejemplo del diagrama de secuencias, en el cual también se definen sus principales símbolos.

2.3. Consistencia

La consistencia pretende que los requisitos de una especificación no se contradigan entre sí [7]. Se emplea entre diagramas UML para lograr que cada uno de ellos hable del mismo problema

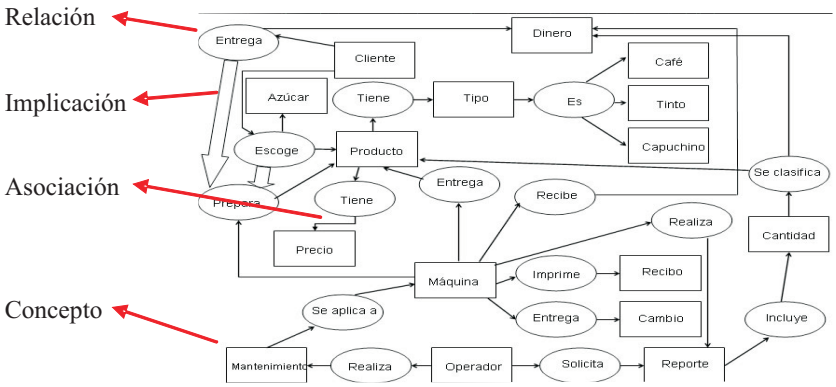


Figura 1. Ejemplo de un esquema preconceptual y su simbología.

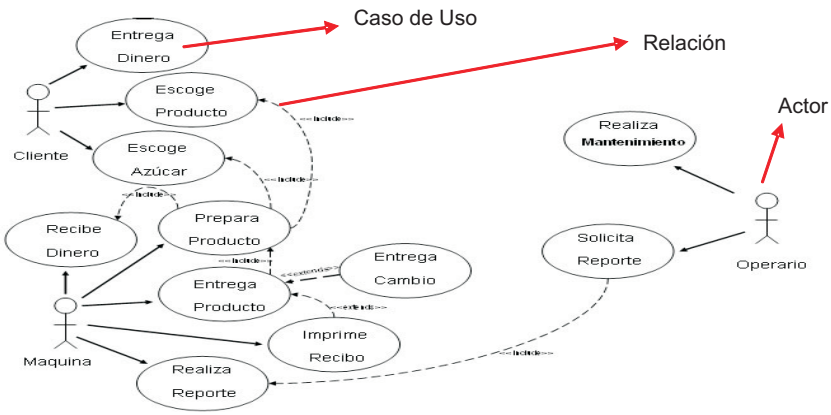


Figura 2. Ejemplo de un diagrama de casos de uso y su simbología.

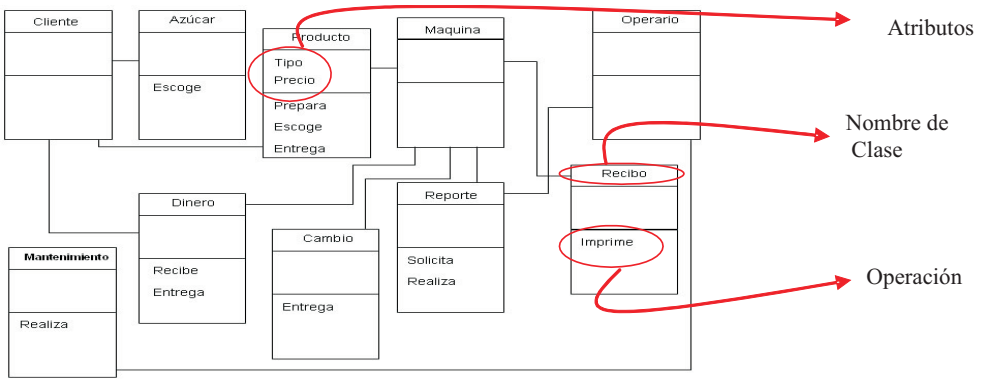


Figura 3. Ejemplo de un diagrama de clases y su simbología.

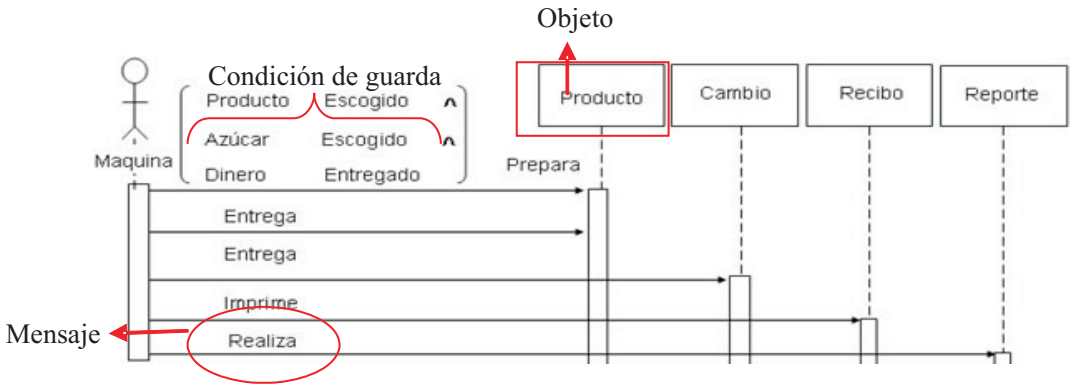


Figura 4.Un ejemplo del diagrama de secuencias y su simbología.

del dominio, puesto que se modela un problema en diferentes diagramas y no un problema distinto en cada diagrama.

A continuación, se enuncian algunas reglas de consistencia entre los diagramas UML y los esquemas preconceptuales [9], y entre los mismos diagramas UML.

- *Un concepto origen de una relación "tiene" o "incluye" de un esquema preconceptual, es una clase candidata del diagrama de clases.*
- *Un concepto destino de una relación "tiene" o "incluye" de un esquema preconceptual, es un atributo candidato, en el diagrama de clases, de la clase candidata que es el concepto origen de la relación.*
- *En un esquema preconceptual, el conjunto de conceptos y relaciones previos a una implicación constituye una condición de guarda candidata en un diagrama de secuencias.*
- *En el nombre de un caso de uso, el concepto (sustantivo) está asociado con una clase del diagrama de clases y la relación (el verbo) está asociada con una operación de esa clase.*
- *Un mensaje del diagrama de secuencia puede representar el verbo del nombre de un caso de uso en el diagrama de casos de uso.*

3. Los Juegos en la Enseñanza de Ingeniería de Software

Wankat y Oreovicz [18] propusieron un conjunto de estrategias didácticas para enseñanza de la Ingeniería en general, entre las que se incluyen las clases magistrales y los proyectos prácticos; otras estrategias, tales como los juegos, los estudios de casos y la educación personalizada, poco se han aplicado en la enseñanza de la Ingeniería de Software. Estos autores, además, afirman que la clase magistral como estrategia didáctica requiere ser complementada por

otras estrategias para alcanzar objetivos cognitivos de más alto nivel, y en esto coinciden con Rugarcia *et al.* [19], quienes sugieren la complementación de la enseñanza tradicional de la Ingeniería con nuevos métodos alternativos que ofrezcan buenas posibilidades de éxito. Los juegos constituyen estrategias que pueden complementar la enseñanza tradicional; no se trata de sustituir las clases magistrales y los proyectos prácticos, sino de suministrar otros espacios que permitan afianzar los conceptos que se imparten en los tipos tradicionales de enseñanza.

Demsey, Ramussen y Luchasen [12] consideran un juego como un formato de aprendizaje que está delimitado por reglas e implica competencia. Bushell [13], por su parte, considera un juego como una actividad interactiva en la cual se recrean condiciones en el mundo real, a fin de estimular el aprendizaje de la toma de decisiones. Kober y Tarca [14] afirman que los juegos de simulación tienen muchas ventajas, ya que permiten que los estudiantes incrementen la motivación, desarrollen comunicación, y el trabajo en grupo. A su vez, Klassen y Willoughby [15] resaltan que los juegos en clase incrementan la velocidad de aprendizaje, aumentan la posibilidad de recordar conceptos, y mejoran la retención de los mismos. Los juegos de simulación, además de introducir al estudiante de manera controlada en el mundo real, permiten que el estudiante asuma un rol definido, que le permita tomar decisiones con seguridad y afrontar las consecuencias que ello implica. Existen juegos de simulación en áreas financieras y administrativas –como el juego de la cerveza– y también existen micromundos como el *beefeater* [16], que introducen al jugador en una situación determinada. En la Ingeniería de Software, se han desarrollado algunos juegos, entre los que se encuentran el Juego de los Requisitos [8] y *Problems and Programmers* [17].

4. Diseño Experimental del Juego de la Consistencia

4.1. Objetivo del juego

Llenar correctamente las plantillas predefinidas de cuatro diagramas (esquema preconceptual, diagrama de clases, diagrama de casos de uso y diagrama de secuencias), correspondientes al modelo verbal de un problema específico. Se debe usar una cantidad también predefinida de palabras para los diagramas correspondientes.

4.2. Hipótesis

Cuando un sujeto experimental se somete a la aplicación de "El juego de la consistencia", puede deducir algunos aspectos relativos a la ingeniería de software y puede clarificar los conceptos en relación con la consistencia entre diferentes diagramas.

4.3. Sujetos experimentales

El juego se puede aplicar a cualquier tipo de persona, tenga o no conocimientos previos de modelado.

4.4. Material experimental

- Modelo verbal: Es un discurso en un lenguaje controlado que representa el manejo de una máquina que fabrica y vende café. El discurso es el siguiente:

  El cliente entrega dinero que es recibido por la máquina.

  El cliente escoge el azúcar y escoge el producto, el cual tiene tres tipos que son: tinto, café y capuchino.

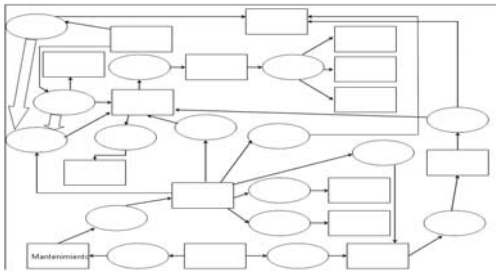
  Cada producto tiene un precio.

  Cuando el cliente haya escogido el producto y el azúcar y cuando haya entregado el dinero, la máquina prepara el producto.

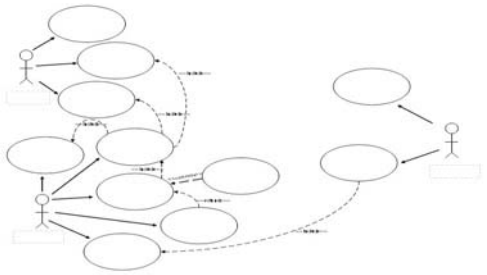
  La máquina entrega el cambio, imprime el recibo y realiza el reporte que incluye la cantidad (clasificada en dinero y producto).

  El operario solicita el reporte y realiza el mantenimiento que se aplica a la máquina.

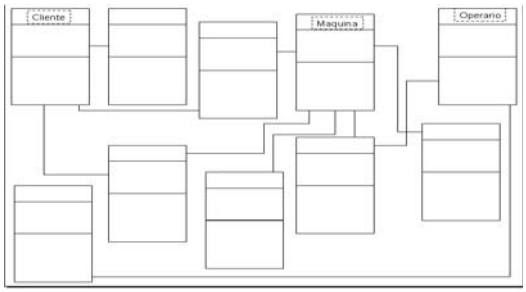
- Plantillas de diagramas: Se pueden apreciar en la Figura 5. Algunas de las plantillas poseen pistas en relación con su elaboración; por ejemplo, en el esquema preconceptual se suministra la ubicación del concepto "mantenimiento" y en el diagrama de clases se incluyen las clases "Cliente", "Máquina" y "Operario". Los otros dos diagramas no presentan pistas porque se pueden deducir de los diagramas mencionados.



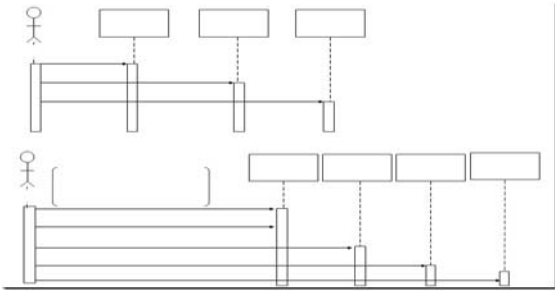
Tablero 1. Esquema preconceptual



Tablero 2. Diagrama de casos de uso



Tablero 3. Diagrama de clases



Tablero 4. Diagrama de secuencia

Figura 5. Tableros de juego.

– Fichas de palabras: Son las palabras del discurso verbal, pero adaptadas de forma que se puedan colocar en los diferentes diagramas.

4.5. Procedimiento

Los jugadores se distribuyen en equipos de cinco a ocho personas.

A cada grupo se le hace entrega de un modelo verbal, cuatro plantillas de diagramas como se muestra en la Figura 5 y las fichas de palabras necesarias para completar los diagramas.

El director del juego anuncia el inicio de un tiempo de 75 minutos, en los cuales los equipos se deben dedicar a completar los diagramas en el orden que deseen. En cualquier momento antes de finalizar el tiempo correspondiente, los equipos pueden entregar sus resultados.

Una vez finalizado el tiempo, el director del juego recoge los diagramas y determina el ganador del juego, comparando los diagramas entregados por los diferentes grupos contra las plantillas mostradas en las Figuras 1 a 4, y asignando los puntajes que se consignan en la Tabla 1. El puntaje máximo es de 123 puntos y, en caso de existir empate en los puntajes, el ganador será quien primero haya entregado los diagramas correspondientes.

4.6. Variables dependientes e independientes

Como una observación importante, se debe notar que las plantillas corresponden a una de las muchas maneras en las cuales se podría traducir el modelo verbal entregado sobre los cuatro diagramas que se solicitan. Sin embargo, por efectos de las limitaciones de tiempo de juego y las dificultades asociadas con el proceso de conteo de puntos y determinación del ganador, se decidió para el juego predefinir las plantillas con una manera específica de modelado; esto también se hace para evitar la subjetividad en la calificación de diferentes modelos si la respuesta no estuviera predefinida. En este caso, las plantillas y el tiempo de juego definido (75 minutos para su elaboración) constituyen las variables independientes. Las variables dependientes que se desea medir son: el mejoramiento en la definición de consistencia y los diferentes aspectos de la Ingeniería de Software que se espera sean deducidos a partir del juego. Las reglas de consistencia esbozadas en la Sección 2.3. no constituyen variables del experimento, por cuanto se pretende que sean adicionales al aprendizaje mismo del juego.

5. Resultados de la Aplicación del Juego

El juego se aplicó sobre 3 grupos, cuyas características se anotan en la Tabla 2. La validación se llevó a cabo empleando una modificación de las encuestas antes y después del juego de inventarios que describen Klassen y Willougby [15]. En esas encuestas se incluía la definición de consistencia y los elementos que se pudieron aprender a partir de esta experiencia, entre otros aspectos. Para evaluar el mejoramiento en la definición de consistencia, se entregaron las encuestas antes y después del juego a un profesor de Ingeniería de Software, quien asignó una nota entre 0 y 5 a todas las definiciones. Como había diferencias notables entre los diferentes grupos, se decidió estudiarlos a todos por separado y luego analizar el desempeño conjunto de todos los grupos.

El grupo 1 tuvo una calificación de 3.0 antes del juego y de 3.8 después del juego; la calificación inicial era previsible, pues el grupo era bastante heterogéneo, con personas de varias disciplinas, algunas poco ligadas con la Ingeniería de Software. El grupo 2 tuvo una calificación de 4.1 antes del juego y de 4.7 después del juego; este grupo era el que tenía más contacto con el concepto de consistencia, si bien lo venían trabajando durante todo el semestre en su curso de Ingeniería de Requisitos. El grupo 3 era el que menos conocía de consistencia, también por las características heterogéneas, pues se trataba de público de una conferencia iberoamericana que, aunque especializada, reunió personas de dife-

Tabla 1
Puntuación del juego de la consistencia

Diagrama	Puntos
Esquema preconceptual	+1 pto por cada elemento correcto.
Casos de uso	+1 pto por cada elemento correcto.
Diagrama de clases	+3 pto por cada clase correcta. +2 pto por cada operación correcta. +1 pto por cada atributo correcto.
Diagrama de secuencias	+1 pto por cada elemento correcto.

Tabla 2
Características de los grupos de participantes en el juego

Código	Semestre	Curso	Nº Participantes	Perfil
1	2006_01	Ingeniería de software, Ingeniería lingüística	6	Grupo combinado de estudiantes y profesores de Ingeniería de Software y Lingüística Computacional
2	2006_03	Ingeniería de requisitos	13	Estudiantes de sexto a noveno semestre de Ingeniería de Sistemas e Informática
3	2007_01	Tutorial: Uso de esquemas preconceptuales para la generación automática de diagramas de clases, comunicación y máquina de estados de UML	41	Grupo combinado de estudiantes y profesores de distintas áreas ligadas con la Ingeniería del Software y la Ingeniería del Conocimiento.

rentes disciplinas; este hecho se vio reflejado en la calificación de 1.5 antes del juego, que se incrementó a 2.7 después del juego. La media ponderada de la calificación antes del juego para los tres grupos fue 2.2 y presentó un incremento del 37% para ubicarse después del juego en 3.1. Si bien no es el ideal de calificación en lo relativo a un concepto como el que se evalúa, las definiciones mostraron una evolución significativa, especialmente tomando en consideración las características heterogéneas de los participantes.

Ahora, en relación con los aspectos susceptibles de deducir a partir del juego, en la Tabla 3 se compendian las principales opiniones acerca de los asuntos que pueden ser considerados como aprendizaje del juego. Para el análisis de estos resultados, se recurrió al modelo de las 4 P's propuesto por Pressman [1] y que toma en consideración cuatro aspectos: Personal, Producto, Proceso y Proyecto. De esta manera, las opiniones expresadas por los participantes en una pregunta abierta (¿Qué fue lo más importante que usted aprendió de este juego?) se totalizaron, se compararon contra el total de participantes y se ligaron con uno de los aspectos de las 4 P's. Se pudo concluir que este puede ser considerado un juego muy ligado con el aspecto "Proceso", aunque se presentaron también opiniones relativas a "Personal". De "Proyecto" y "Producto" no hubo opiniones, y era de esperarse por la ubicación de este juego en el proceso inicial de captura de requisitos.

Es importante señalar que los seis aprendizajes señalados hacen parte de los tópicos que se pueden enseñar de manera tradicional en los cursos de Ingeniería de Software.

Finalmente, el 100% de los participantes contestó que esta fue una experiencia de aprendizaje que valió la pena. Algunas de las explicaciones que dieron para ello se consignan en la Tabla 4. Los porcentajes son bajos porque la mayoría de los participantes se limitó a responder afir-

Tabla 3
Resumen de las opiniones en relación con los aspectos de aprendizaje del juego de la consistencia

Aprendizajes	%	Aspecto
Cómo y cuando se valida la consistencia entre diagramas	48	Proceso
Utilidad de los diferentes diagramas	45	Proceso
Fases del desarrollo de software	27	Proceso
Comunicación en el grupo de trabajo	13	Personal
Trabajo en equipo	7	Personal
Importancia de la documentación y el modelado	6	Proceso

Tabla 4
Razones para creer que el juego de la consistencia es una experiencia de aprendizaje que valió la pena

Razones	%
Secuencia en la realización de diagramas	13
Los esquemas preconceptuales ayudan a manejar la consistencia	8
El juego fue una experiencia de aprendizaje	7
Transformación entre diagramas	5
Parece simplificar la consistencia	5
La consistencia a veces se pasa por alto en la realidad	2
En la realidad existe mucha presión para el modelado	2

mativamente a la pregunta, sin aducir ninguna razón para ello. Aparecen, también en esta Tabla, elementos que podrían hacer parte de cualquier curso de Ingeniería de Software, como la transformación entre diagramas o la secuencia en la realización de diagramas.

Cabe anotar que los participantes manifestaron de forma mayoritaria (92% de las opiniones) no haber leído libros sobre consistencia y los que habían leído materiales de clase sobre consistencia, UML o grafos conceptuales fueron una minoría (37%).

6. Conclusiones

Entre las principales conclusiones de este trabajo se cuentan las siguientes:

- Para la enseñanza de la Ingeniería de Software se han empleado diversas estrategias tradicionales, que poco se han complementado con otras experiencias que no se centren en el docente como responsable de enseñar conceptos. Esto ha generado la búsqueda de alternativas de enseñanza como los juegos; para ello, ya se han comenzado a generar algunos trabajos al respecto.
- El Juego de la Consistencia es una herramienta didáctica para usar en las aulas de clase, que le permite al estudiante afianzar conocimientos sobre modelado, métodos de desarrollo de software, trabajo en equipo y comunicación y, sobre todo, consistencia entre diagramas.
- El Juego de la Consistencia se ha practicado con grupos heterogéneos de estudiantes y profesores y se ha llegado a resultados similares en cuanto a los conocimientos que se afianzan mediante su práctica: se ha mejorado la definición de consistencia y se han confirmado algunos de los aspectos fundamentales en el proceso de desarrollo de software, especialmente en las etapas de modelado.
- La práctica del Juego de la Consistencia ha mostrado, finalmente, que los juegos en clase no reemplazan la enseñanza tradicional de la Ingeniería de Software, sino que la complementan. La existencia de conocimientos previos en relación con los temas que se abordan en el juego ha demostrado ser fundamental para las conclusiones a que llegan sus participantes. Sin embargo, algunos conceptos del manejo de proyectos, tales como el trabajo en equipo y la importancia de la comunicación entre los integrantes del grupo son difíciles de enseñar mediante métodos tradicionales; la experimentación de situaciones como las que aborda el juego permite que esos conceptos se refuercen, de forma que se puedan incorporar en el aprendizaje significativo de los practicantes.

7. Trabajo Futuro

Con base en esta experiencia, se propone la realización de las siguientes actividades:

- Aplicación de "El Juego de la Consistencia" en grupos de personas con diferente perfil a los hasta ahora abordados. Sería especialmente importante conocer las reacciones frente al juego de profesionales de la Ingeniería de Software y de Sistemas en general. También se podría realizar el juego en personas con pocos conocimientos en modelado o consistencia.
- Incorporación de otras variables en el juego, como por ejemplo un mecanismo que permita mayor libertad a los participantes, que no se limite únicamente al llenado de los espacios en blanco, sino que puedan hacer propuestas concretas de modelado.
- Elaboración de otros juegos que permitan mejorar o complementar "El Juego de la Consistencia", u otros temas como Aspectos, Puntos de Vista o Refinamiento, de gran actualidad en la Ingeniería de Software.

Referencias

1. Pressman R.: "Ingeniería del Software: un Enfoque Práctico", McGraw-Hill, Madrid, 2002.
2. Zelkowitz M.V., Shaw A. C., and Gannon J. D.: "Principles of software engineering and design", Prentice Hall, Englewoods Cliffs, 1979.
3. Bohem B.: "Software Engineering". IEEE Transactions on Computers, C-25, No. 12 (1976) 1226–1241.

4. Bauer, F.L.: "Software Engineering". In: Proceedings of the IFIP Congress 71, ed.Friedman, C.V., North Holland (1972), 530–538.
5. Rumbaugh J., Blaha M., Premerlani W., Eddy F., and Lorensen W.: "Object-Oriented Modeling and design", Prentice Hall, New Jersey. 1991
6. OMG. OMG Unified Modeling Language Specification. Object Management Group. Available: http://www.omg.org/UML/. [Citado 13 de Diciembre de 2005]
7. Zowghi D. and Gervasi V.: "The Three Cs of Requirements: Consistency, Completeness and Correctness". In: Proceedings of 8th International Workshop on Requirements Engineering: Foundation for Software Quality, (REFSQ'02), Essen, Germany (2002).
8. Zapata C. M. y Awad G.: "El Juego de los Requisitos: Enseñanza de la Gestión de Proyectos de Software". En: Memorias del XIII Congreso Iberoamericano de Educación Superior en Computación CIESC2005, Cali (2005), 33-43.
9. Zapata C. M., Gelbukh, A. y Arango, F.: "Pre-conceptual Schema: A Conceptual-Graph-Like Knowledge Representation for Requirements Elicitation", Lecture Notes in Computer Science, Vol. 4293 (2006) 17-27.
10. Sowa J. F.: "Conceptual Structures: Information Processing in Mind and Machine", Addison-Wesley Publishing Co., 1984.
11. Booch G.: "El proceso Unificado de Desarrollo de Software", Addison Wesley, Madrid, 2001.
12. Dempsey J. V., Rasmussen K. and Lucassen, B.: "The Instructional Gaming Literature: Implications and 99 Sources". COE Technical Report No. 96-1, College of Education, University of South Alabama, 1996.
13. Bushell T.: "The role of the business game in management education". In: Proceedings of the Conference Reflections on Teaching: Maintaining Quality in Changing Times, Low Wood, Lake Windermer (2001).
14. Kober R., and Tarca A.: "For fun or profit? An evaluation of a business simulation game". Accounting Research Journal, Vol. 15 (2000) 98-111.
15. Klassen K. y Willoughby K.: "In-Class Simulation Games: Assessing Student Learning". Journal of Information Technology Education, Vol. 2, (2003), 1-13.
16. Strategy Dynamics: "Beefeater restaurants microworld". Available http://www.strategydynamics.com/products/beefspec1.asp [Citado 13 de Dic. de 2005]
17. Baker A., Navarro E., and Van der Hoek A.: "An experimental card game for teaching software engineering processes". The Journal of Systems and Software, No. 75, (2005), 3–16.
18. Wankat, P.C. y Oreovicz, F.S.: "Teaching Engineering", McGraw-Hill, Nueva York, 1993.
19. Rugarcia, A., Felder, R., Woods, D. y Stice, J.: "The Future of Engineering Education I: The vision for a new century". Chemical Engineering Education, Vol. 34, No. 1, (2000), 16-25.

Recibido el 02 de Octubre de 2006
En forma revisada el 11 de Febrero de 2008