PoV-GAME: PUNTOS DE VISTA MEDIANTE JUEGOS*

Carlos Zapata-Jaramillo**
Guillermo González-Calderón***
Darliana Rivera****

Recibido: 10/06/2011
Aceptado: 24/04/2012

RESUMEN

El desarrollo de proyectos de software considera importante la consistencia entre la información entregada por todos los interesados de una aplicación. En otras palabras, se procura llegar a acuerdos entre los diferentes puntos de vista de cada actor y llevarlos a cabo durante todo el proyecto. Con los métodos de enseñanza tradicionales se procura formar habilidades en este campo, pero no se afianzan mediante la práctica que, generalmente, se emplea para el aprendizaje en este campo. Por ello, en este artículo se propone "PoV-GAME", una nueva forma de "jugar aprendiendo", la cual busca fortalecer algunos conceptos básicos en la ingeniería de software como la consistencia en la información que se maneja a lo largo de un proyecto y afianzar la importancia de los puntos de vista en el desarrollo de un producto de software. Posteriormente, se analizan los resultados obtenidos luego de realizar el juego en diferentes grupos de estudiantes.

Palabras clave: juegos instructivos, desarrollo de software, aprendizaje, puntos de vista, trazabilidad.

* Este Artículo es resultado del trabajo de grado de Darliana Rivera, en el marco del proyecto de investigación "PoV-GAME: PUNTOS DE VISTA MEDIANTE JUEGOS", desarrollado en la Universidad Nacional de Colombia.

** Ph. D en Ingeniería; Profesor asociado de la Universidad Nacional de Colombia; Facultad de Minas, Escuela de Sistemas; E-mail: cmzapata@unal.edu.co.

*** Ph. D(C) en Ingeniería; Universidad Nacional de Colombia; Facultad de Minas, Escuela de Sistemas; Universidad de Medellín. E-mail: ggonzal@unal.edu.co.

**** Ingeniera de Sistemas e Informática; Universidad Nacional de Colombia; E-mail: darivera@unal.edu.co.

PoV-GAME: VIEWPOINTS THROUGH GAMES

ABSTRACT

Information for software applications is delivered by the stakeholders. Software development projects need consistency as an important aspect of such information. In other words, agreement among the different viewpoints must be reached in order to be implemented in the project. Well-known teaching methods try to create capabilities in this field, but they fail in reinforcing them in practice, which is commonly used for teaching in this field. Therefore, in this paper we propose "PoV GAME" a new way of "play-to-learn". We aim the reinforcement of some basic concepts about software engineering such as consistency in the information handled throughout the project, and the importance of viewpoints in the software product development. Subsequently, we analyze the results obtained after holding the game in several groups of students.

Key words: educational games, software development, learning, viewpoints, traceability.

INTRODUCCIÓN

En el desarrollo de aplicaciones de software es necesario tener en cuenta los puntos de vista de los diferentes actores del sistema. Esto se hace para lograr que el proyecto final sea consistente con todos los entes que componen la organización [1]. Así, es necesario que los ingenieros de software desarrollen habilidades para identificar estos puntos de vista [2].

Tradicionalmente, la ingeniería de software, y en particular los puntos de vista, se enseñan usando métodos comunes como las llamadas clases expositivas y los proyectos prácticos "de juguete" [3]. En el primer método, el docente es el encargado de contar a los alumnos toda la información sobre un tema determinado. En el segundo, los alumnos, con asistencia del docente, realizan una simulación práctica de los conceptos adquiridos. Estos métodos no garantizan el desarrollo de las habilidades que requieren los ingenieros de software.

Como una respuesta a esta limitación, se vienen empleando métodos interactivos para la enseñanza, tales como los juegos. Existen trabajos en juegos para administración [4] y negociación [5]. En ingeniería de software también se encuentran algunos ejemplos enfocados en aspectos como la labor de los programadores [3], la captura de requisitos [6, 7], la consistencia entre diferentes diagramas [8], la construcción de software [9] y el manejo de riesgos [10], pero, en general, no contribuyen a generar conciencia en los equipos de trabajo sobre la importancia de los puntos de vista en el desarrollo de aplicaciones.

En este artículo se presenta PoV-GAME, una estrategia didáctica que pretende afianzar el conocimiento adquirido acerca de los puntos de vista en el desarrollo de aplicaciones de software. Este es un juego de cartas en el cual se pretende que los jugadores procesen el pago de la nómina al empleado de una compañía. Es indispensable contar con un orden lógico y tener en cuenta las etapas del proceso en las diferentes dependencias de la compañía.

Este artículo se organiza así: en la sección 1 se discute el marco teórico que fundamenta la propuesta; en la sección 2 se revisan de manera crítica los antecedentes del proyecto; en la sección 3 se propone PoV-GAME; en la sección 4 se realiza el análisis de resultados y, finalmente, en las secciones 5 y 6 se discuten las conclusiones y el trabajo futuro.

1 MARCO TEÓRICO

1.1 Puntos de Vista

La ingeniería de software es una ciencia de la computación basada en lógica matemática y modelos computacionales. La primera fase de la fabricación de una aplicación involucra la toma de importantes decisiones por parte del analista de requisitos [11].

Dentro de la gestión de un proyecto es importante contar con los puntos de vista de los diferentes actores que componen la organización. Un punto de vista (PoV) es la combinación de un actor, una fuente de conocimiento y un rol dentro de un proceso de desarrollo y la idea que el actor mantenga sobre la perspectiva del proyecto, el cual encierra un conocimiento parcial del sistema y del dominio en particular [1, 2]. Cada especificación debe ser coherente con todas las otras especificaciones detectadas [12].

El diseño y modelado de soluciones a problemas independientes (que cuentan con un solo actor) no se encuentra con frecuencia en ingeniería de software. Por el contrario, son comunes aquellos proyectos que suelen contar con diversos interesados para la misma aplicación, lo cual eleva considerablemente la complejidad [12].

Dado que es un proceso que la mayoría de las veces se torna difícil, es necesario tener estrategias que permitan comprender bien qué son los puntos de vista y cómo se aplican en un proyecto. El análisis de este tema, particularmente, busca que los futuros ingenieros de software puedan aprovechar las bondades de este tipo de conceptos que muy pocas veces se tratan en un proyecto en desarrollo [13].

Los métodos convencionales de desarrollo no le dan la debida importancia a este tema y, en su lugar, proponen estructuras rígidas y controles a la diversidad de opiniones y sus relaciones. La importancia del uso de los puntos de vista radica en el conocimiento de cualquier aspecto de la aplicación, la terminología adecuada y el conocimiento de las necesidades del interesado para encontrar un equilibrio entre las perspectivas, logrando coherencia en el trabajo en grupo [2]. La consistencia se puede lograr integrando las soluciones parciales encontradas en los diferentes actores.

El principal objetivo de los puntos de vista en la educción de un proceso es comparar diferentes enfoques y proporcionar un sistema de negociación de conflictos [11]. Existen dificultades importantes asociadas con los puntos de vista: su identificación, la gestión de información del enfoque en particular y la elección del modelo apropiado para tratarlos. La primera dificultad se podría considerar un riesgo, ya que se podría llegar a encontrar una cantidad innecesaria de información. Para ello, es importante clasificarla partiendo del dominio y alcance del proyecto como requisito fundamental [2].

La múltiple información conseguida se puede presentar como una ventaja y desventaja al mismo tiempo. La ventaja radica en que es menos posible que falte información crítica del sistema, pero es bastante difícil gestionar esa información y eliminar la redundancia o contradicciones que se puedan llegar a generar y, por tanto, podría, fácilmente, convertirse en una desventaja. Además, en el mercado no existe ninguna herramienta comercial disponible que ayude con esta selección [2].

1.2 Métodos tradicionales de enseñanza de la ingeniería de software

Diferentes planes de estudio abarcan algunos de los conceptos fundamentales en ingeniería de software. Entre ellos, es significativo rescatar la gestión de habilidades de comunicación. Su importancia radica en que la ingeniería de software es una actividad basada en el trabajo de las personas. Por ello, la gestión y la comunicación deben ser constantes [14].

Algunos de los temas importantes que debe tratar la ingeniería de software son aquellos que involucran la parte económica, los métodos, las herramientas técnicas para todas sus etapas (definición, análisis, diseño, implementación, validación y mantenimiento), las habilidades de comunicación interpersonal, el conocimiento de las necesidades de los clientes y la terminología, entre otros. [15].

La enseñanza de conceptos generalizados y tradicionales de la ingeniería misma satura los planes de estudio relacionados con la ingeniería de software. Conceptos tales como la solución creativa de problemas, la fiabilidad, el mantenimiento, la documentación de diseño, la economía y la calidad, entre otras, comprenden los temas en la ingeniería de software. El problema que enfrentan los educadores en ingeniería de software es el diseño de cursos que asignen la debida importancia a estos valores. Muchos de estos conceptos se pueden plantear pero es necesario reforzarlos durante todo programa académico [15]. Muchas profesiones siguen el modelo tradicional en sus planes académicos, pero este modelo no parece adecuado para la ingeniería de software, principalmente por ser una disciplina bastante compleja [14, 16].

La enseñanza tradicional de la ingeniería de software tiene en cuenta las funciones de cada uno de los actores dentro de un proceso específico. Las técnicas pedagógicas que comúnmente se usan para ello son las clases expositivas y los proyectos prácticos ("de juguete") [3]. Allí, los alumnos alcanzan únicamente el aprendizaje de algunos conceptos elementales, pero no se acercan a las situaciones reales del proceso de educción de requisitos.

Los ingenieros de software también deben tener conocimiento de la aplicación particular. Se debe tener conocimiento de las necesidades de los

clientes, la terminología y las prácticas en el área de aplicación, las cuales son tan importantes como las capacidades de desarrollo de la ingeniería de software [14]. Por ello, los centros de estudio de educación superior buscan proyectar a los estudiantes el escenario ideal para la enseñanza de la ingeniería de software. Este escenario incluye la profundización en temas como: ciencias de la computación, matemáticas, ciencias sociales, empresariales, de gestión y un área de aplicación. Sin embargo, algunas veces este proyecto se queda únicamente siendo un escenario ideal, ya que típicamente el ingeniero de software tiene poca o ninguna comprensión de diseño, análisis de requisitos, especificación funcional, planificación, configuración de control o de garantía de calidad [15]. Además, los nuevos graduados no le dan importancia a la economía en el software, nunca estiman el tamaño o el calendario de un proyecto de software y probablemente nunca participan en una revisión del diseño o el código. Sin embargo, todos estos ítems son fundamentales para la ingeniería de software [15].

Los métodos de enseñanza no tradicional, como los juegos y su aplicación en la ingeniería de software, pretenden cambiar este paradigma y lograr acercarse más al modelo ideal, mezclando el conocimiento adquirido en las ciencias de la computación con experiencias vivenciales de simulación [3].

2 ANTECEDENTES

2.1 Uso de juegos en el aprendizaje

Los juegos se vienen utilizando para ampliar el nivel de aprendizaje en diferentes disciplinas. Un ejemplo de ello es "El juego de la cerveza" [4], el cual permite afianzar distintas áreas de un modelo sistémico aplicadas a la administración como la teoría de sistemas, la gerencia y la información, logrando ver cómo el punto de vista individual interactúa con el sistema en general. El juego de la cerveza señala que la función de cada actor dentro de una organización consiste en "administrar su posición" de forma aislada [4].

Gardner [5] también implementó los juegos como una experiencia de aprendizaje para los estudiantes de administración de empresas y economía, mediante juegos de estrategia. Entre ellos se encuentra "El Juego de la negociación", el cual presenta múltiples formas de negociación: secuencial, de ofertas y contraofertas, rechazo de ofertas, entre otras, logrando plantear un contexto "real" en el que las oportunidades de cooperación son de gran importancia en la negociación [5].

2.2 Los juegos como métodos de enseñanza de la ingeniería de software

En la actualidad, se viene ejecutando una serie de juegos que permite incrementar el nivel de aprendizaje de las ciencias, entre las que se incluye la ingeniería de software y el desarrollo de aplicaciones [3]. Dentro de la ingeniería de software es posible encontrar varios de ellos, que se analizan en esta sección.

Un primer ejemplo lo constituyen problemas y programadores, un juego de cartas que procura simular el proceso de desarrollo de una aplicación, tomando en consideración las diferentes situaciones que se pueden presentar a un programador en ese proceso. Con el juego se señala la importancia de la acertada toma de decisiones de parte del grupo de trabajo y se puntualiza la importancia de las diferentes fases del desarrollo de software [3]. Un segundo juego, la Consistencia, el cual se fundamenta en completar adecuadamente, y con determinado número de palabras, las plantillas de cuatro diagramas que representan una de las posibles soluciones del modelo verbal de un problema concreto [8]. Otro de los juegos importantes de mencionar es el juego de los requisitos, encargado de simular, de forma práctica, las condiciones de desarrollo en un entorno competitivo similar al presentado en la vida cotidiana. Cada miembro del equipo tiene asignadas unas funciones y

debe seguir determinadas reglas para cumplir su papel [6].

Otro juego, conocido como el juego del diálogo de educción de requisitos, busca lograr que los jugadores se conciencien de la importancia del diálogo adecuado en la educción de requisitos y su posterior transformación a esquemas preconceptuales [7].

En este dominio, también es posible encontrar el juego del desarrollo de software, cuyo objetivo es construir cajas de origami marcadas con uno de los siguientes grupos de letras: SO, FT, WA o RE, cada una de las cuales representa un módulo. Todos los grupos compiten para obtener mayores beneficios para su empresa imaginaria [9].

Por último, es importante mencionar el manejo de riesgos, que es un juego de mesa desarrollado en la Universidad Carnegie Mellon para un programa de ingeniería, especializado en enseñar conceptos de gestión de riesgos. Cada jugador asume el rol de director de proyecto y todos compiten unos contra otros para lograr desarrollar un producto, venderlo en el mercado y finalmente ganar más dinero para derrotar los competidores. El objetivo del juego es mejorar el aprendizaje y la práctica de toma de decisiones a través de la simulación de un proyecto de desarrollo de software [10].

Todos estos juegos se aplicaron en grupos con características diferentes en cuanto a nivel de escolaridad y cantidad de participantes, entre otras variaciones, y se obtuvieron resultados muy positivos, entre ellos el aprendizaje y la importancia del trabajo en equipo, la comunicación entre los integrantes del grupo, el trabajo bajo presión, la comunicación permanente con el cliente, la división de tareas por actividades, las características de un buen director, la motivación, el seguimiento en el desarrollo de un proyecto, la toma de decisiones, el propósito de la ingeniería de software, las fases del desarrollo de software y la necesidad de lograr consistencia entre diagramas, entre otras. Sin embargo, los juegos analizados no enfatizan en la importancia de los puntos de vista en el desarrollo de software y, particularmente, en la manera en que se entremezclan para lograr una comprensión adecuada del proceso que conduce a una aplicación. Por ello, en la siguiente sección se presenta PoV-GAME, para solucionar estas limitaciones.

3 PoV-GAME: ENSEÑANZA DE LOS PUNTOS DE VISTA DE UNA APLICACIÓN MEDIANTE UN JUEGO DE CARTAS

3.1 Objetivo del juego

El objetivo que persigue el juego es concienciar a los participantes sobre la importancia de llegar a acuerdos entre los diferentes puntos de vista de los actores de una aplicación. Así, se busca resaltar el resultado que se genera al unir las funciones de cada uno de los actores de un proyecto de manera consistente, como una de las infinitas soluciones informáticas que tiene un problema. Además, se procura reconocer el alcance de una aplicación y seleccionar la información apropiada con el fin de lograr la satisfacción de los interesados.

El juego se dirige, principalmente, a aprendices de ingeniería de software, aunque se puede jugar con cualquier tipo de personas que quieran aprender la importancia de los puntos de vista en el desarrollo de una aplicación.

3.2 Materiales

Para la realización del juego es necesario contar con los siguientes materiales:

- 129 cartas del juego, entre las que se encuentran comodines, actores, objetos y procesos. Algunas de estas cartas se ejemplifican en la figura 1.
- Un tablero de juego, que se muestra en la figura 2, donde se colocan las cartas en un orden específico, que representa el pago de nómina de una compañía.

3.3 Procedimiento

El juego consiste en armar correctamente nueve frases que componen algunos de los procesos

que se deben llevar a cabo a la hora de realizar el pago de la nómina en una compañía. Al inicio, el director del juego se encargará de extender sobre una superficie plana el tablero de juego, ubicando los jugadores de tal forma que todos puedan leer el contenido de las casillas del tablero.

El juego permite de 2 a 4 jugadores. Cada jugador saca de la baraja una carta elegida al azar con el fin de determinar el orden de los participantes en el juego. Quien saque el mayor puntaje en la carta obtenida será quien inicie y se seguirá en orden como lo indican las manecillas del reloj.

Posteriormente, el coordinador del juego repartirá a cada participante 12 cartas elegidas al azar. Cuando todos tengan sus cartas, se darán 3 minutos para organizar el juego.

Terminado este tiempo, el primer jugador sacará una carta adicional de la baraja (este movimiento se denomina en adelante "arrastre"), verificará si puede poner sus cartas en juego (este movimiento se denomina en adelante "bajarse", según se especifica en la sección 3.4) y luego lanzará una carta a su compañero de la derecha (este movimiento se denomina en adelante "tirar"); este decide si la recoge o si prefiere "arrastrar" de la baraja y continúa el juego como se explicó anteriormente.

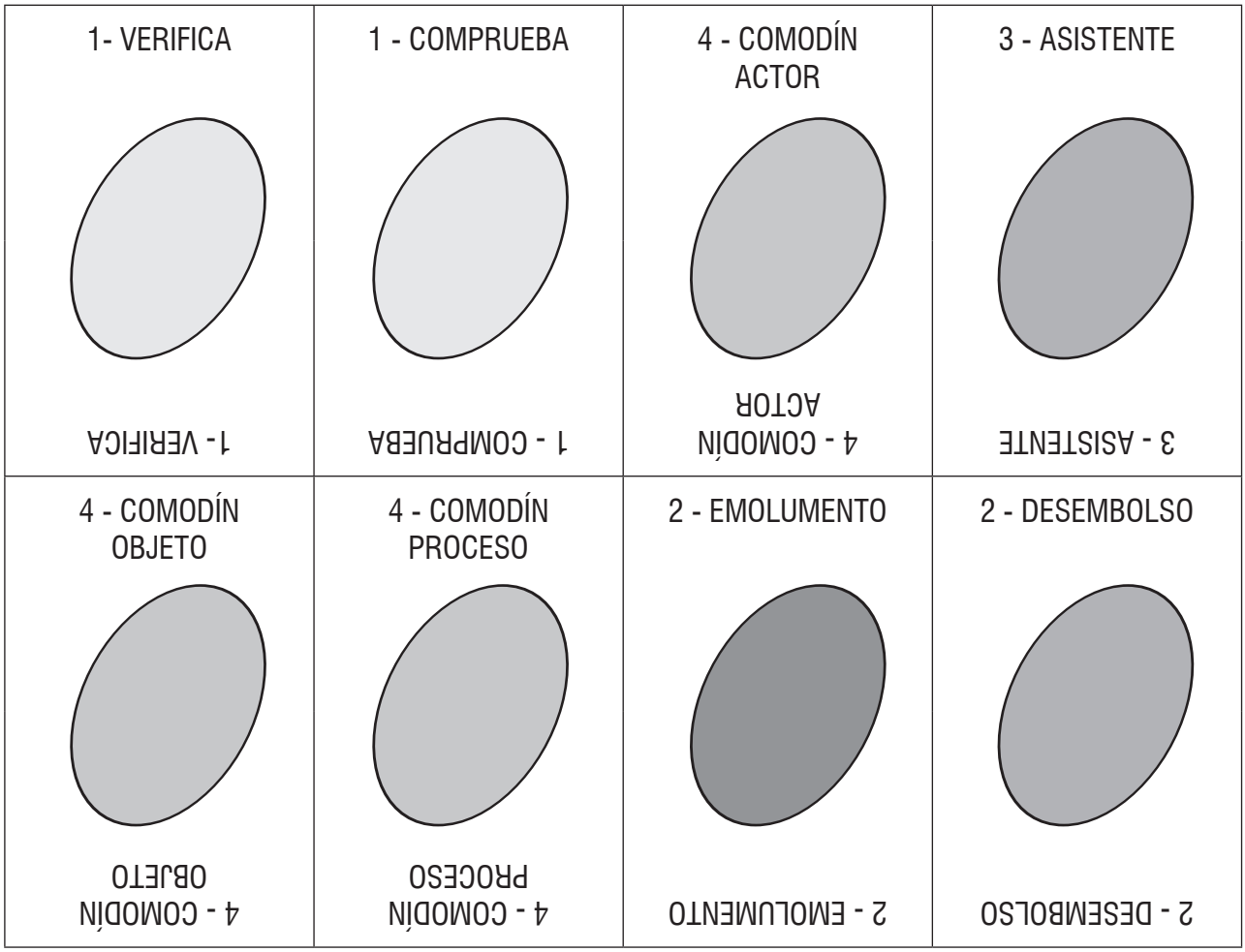

Figura 1. Ejemplos de cartas de juego

Fuente: elaboración propia.

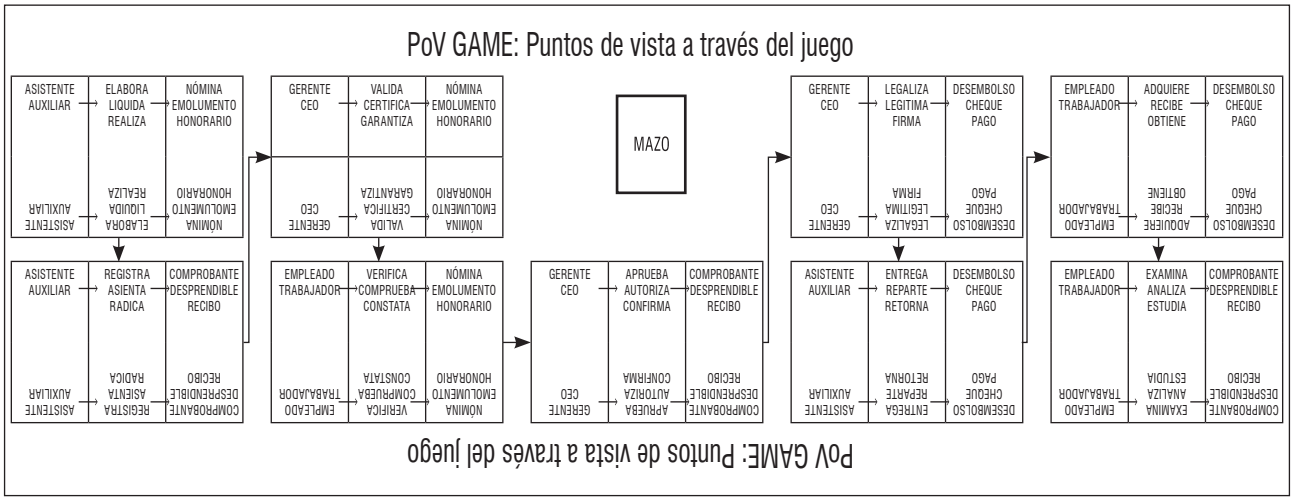

Figura 2. Tablero de juego

Fuente: elaboración propia.

3.4 Reglas

Existe una serie de restricciones generales y unas normas que se deben tener en cuenta a la hora jugar, principalmente al armar las ternas (conjuntos de tres cartas del tipo actor-proceso-objeto).

- Para colocar las cartas en el tablero de juego, el participante debe estar en su turno. Las cartas se ponen boca arriba y en el lugar que le corresponde del tablero. Cada terna se podrá armar solo una vez, es decir, en cada casilla solo se podrá colocar una carta.
- El tablero de juego contiene, en cada casilla, un listado de palabras que se podrán colocar allí. Los procesos pertenecen a un actor específico y se realizan sobre un objeto determinado.
- Siempre que un jugador coloque sus cartas en juego, deberá hacerlo con ternas completas. La primera vez deberán ser mínimo dos oraciones; las veces siguientes podrá bajar desde una terna completa.
- Se permite el cambio de comodines por la palabra específica de la casilla, siempre y cuando el comodín continúe en juego inmediatamente. Esto no aplica a la hora de bajarse por primera vez.
- Existen tres tipos de comodín (actor, proceso, objeto), los cuales se pueden usar solo en la casilla correspondiente al rol específico en cada uno. Se puede usar máximo un comodín por oración.
- Solo se puede recoger la última carta "tirada" por el participante de la izquierda y ninguna otra. En caso de no elegir esta carta, se podrá arrastrar una de la baraja.
- Debe existir consistencia entre los nombres asignados a cada actor y objeto en el tablero. En otras palabras, el nombre otorgado desde el inicio del juego a cada rol y objeto deberá ser el mismo durante todo el transcurso del juego.

3.5 El ganador

Para determinar el ganador se utilizan varias técnicas, pero la primordial es que el participante que "baje" primero todas las cartas propias será el vencedor de la partida. En caso que se genere un empate o que se terminen las ternas sin que ninguno de los jugadores quede sin cartas, ganará el jugador que menos cartas tenga en su poder. Si aun así continúa el empate ganará el jugador que menos puntaje total tenga luego de sumar el puntaje de cada carta. La última forma de encontrar el ganador, en caso de que todas las anteriores hayan

arrojado empates, será la misma forma con la que se elige la persona que inicia el juego.

4 RESULTADOS DEL JUEGO

El juego se practicó con seis grupos con características similares. Uno de los grupos conformado por cinco estudiantes de ingeniería de sistemas e informática en diferentes semestres, superiores al quinto. El resto de los grupos, conformados por cuatro estudiantes con las mismas características.

Inicialmente, a cada grupo se le presentó una explicación general de los objetivos del juego, sus reglas, las especificaciones para determinar el ganador y la metodología a usar.

Al terminar el juego, en cada grupo se realizó la validación de los resultados, presentando una encuesta con tres preguntas de respuesta abierta, que permitió hacer el presente análisis. Las preguntas usadas fueron las siguientes:

¿Qué aprendió del juego?, ¿Qué cree que se necesita para ganar? ¿Qué mejoras le haría al juego?

Las respuestas se agruparon por categorías más generales que se pueden verificar en las figuras 3, 4 y 5. En la figura 3 están las categorías de las respuestas de la primera pregunta, con sus respectivos porcentajes de ocurrencia. Los resultados estuvieron muy distantes: la respuesta más aludida por los participantes (el 40% de ellos) fue la importancia de la trazabilidad en el desarrollo de un proyecto de software. Esta respuesta reflejó la mayor preocupación de los participantes en el juego, que era buscar los términos adecuados para lograr conformar una terna completa.

La siguiente respuesta que obtuvo mayor porcentaje fue la especificación de funciones dependiendo del rol asignado (con un 28% de participación). Cada terna del juego permite ver que un rol cumple un papel específico sobre un objeto determinado.

En cuanto a la segunda pregunta, cuyas categorías se pueden observar en la figura 4, la respuesta más citada se enfocó en la necesidad de tener definida una estrategia de juego para ganar la partida (con el 88% de los resultados). Durante el desarrollo del juego, fue evidente que cada jugador adquirió su propia estrategia. Esta variaba con el fin de adaptarse a la etapa del juego en la que se encontrara. Otra respuesta representativa fue la importancia de la suerte en el desarrollo de un proyecto de software (con el 40% de incidencia). Esta se reflejó en los comodines y en las cartas necesarias para completar una terna antes que sus contrincantes. La identificación de las cartas en juego (con el 24%) fue otro ítem importante en este estudio. Aquí, ubicar la posición de las cartas fue fundamental en el transcurso del juego. En

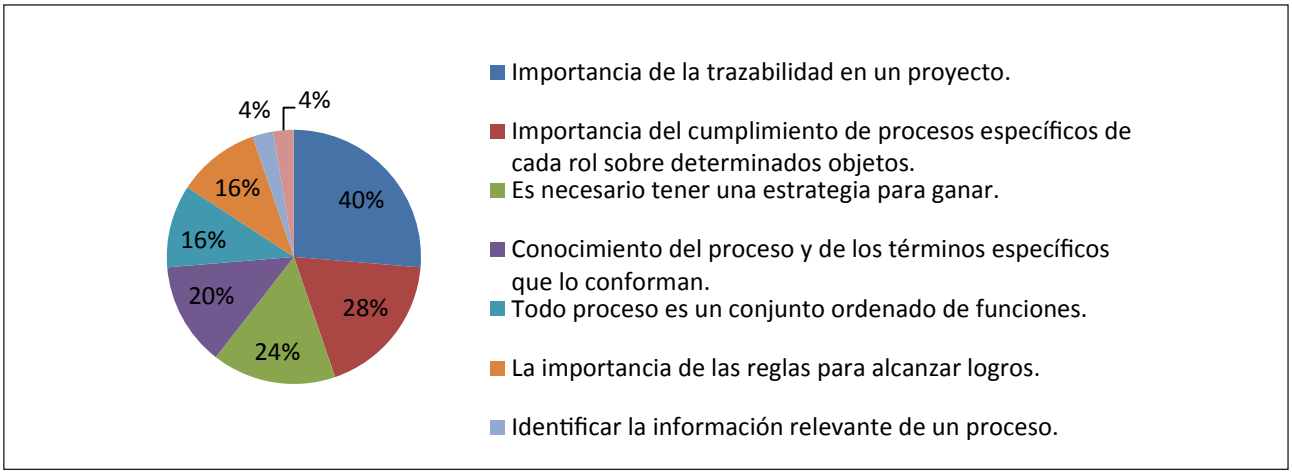

Figura 3. Categorías y porcentajes del aprendizaje del juego

Fuente: elaboración propia.

esta consulta también es notable el desfase en la proporción de los resultados. En el interrogante sobre las mejoras sugeridas al juego, los porcentajes se vieron mucho más divididos que en los dos casos anteriores como se puede evidenciar en la figura 5. De igual forma, las dos categorías que se destacaron (con un 24% de ocurrencia) fueron lograr una mayor identificación de las cartas por categorías y no realizar ningún cambio.

Luego de establecer un porcentaje de respuestas obtenidas en las prácticas del juego, se identificaron nuevamente los objetivos iniciales del juego y se agruparon las respuestas obtenidas en cuatro categorías presentadas en la figura 6.

Allí, es posible notar que el 72% de los participantes logró aprender que el resultado del desarrollo de un producto de software depende de los puntos de vista de los actores. De igual forma, ninguno de los practicantes del juego identificó que el resultado del desarrollo del software no es único, por lo que será necesario replantear este punto para subsecuentes prácticas del juego.

5 CONCLUSIONES

PoV-Game, como enseñanza de los puntos de vista a través de un juego, permite generar conciencia entre sus participantes sobre la importancia que tiene el uso de puntos de vista en los proyectos de software. Esto es posible validarlo con los resultados obtenidos en los seis grupos de muestra tomados para este artículo, quienes lograron una muy buena aproximación al objetivo principal del juego.

La enseñanza de algunas competencias, complementada con ayudas didácticas, como los juegos, hace que los estudiantes de ingeniería de software reconozcan con mayor claridad algunos planteamientos de esta profesión y sea mucho más fácil aplicarlos e interpretarlos en la práctica. Los juegos deben tener unas características especiales para complementar el aprendizaje de una manera eficiente, pues no cualquier actividad refleja los resultados necesarios ni logra los propósitos esperados.

6 TRABAJO FUTURO

Luego de realizar un análisis de los resultados obtenidos en las consultas a los participantes, es posible identificar varios factores importantes para complementar este y otros trabajos similares. A continuación, se mencionan algunos de ellos:

- Explorar los diferentes campos en los que se puede profundizar el aprendizaje mediante enseñanza didáctica, dado que es una opción interesante para lograr que los estudiantes aprendan a aplicar los conceptos necesarios en las buenas prácticas de la ingeniería de software.
- Implementar juegos en otras temáticas de ingeniería de software entre las que se encuentran los ciclos de vida de un proyecto de software y temas especializados como aspectos o pruebas de software. Con esto se pretende desarrollar mayor agilidad en los ingenieros de software para entender y gestionar proyectos prácticos reales con mayor eficiencia.
- Crear una aplicación de software para implementar los juegos desarrollados con el fin de que se pueda jugar en línea y sean accesibles desde cualquier lugar.

REFERENCIAS

[1] A. Finkelstein *et al.*, "Viewpoints: A framework for integrating multiple perspectives in system development," *Intl. Journal on Software Engineering and Knowledge Engineering,* vol. 2, pp. 31-58, 1992.

[2] A. Finkelstein, y I. Sommerville, "The Viewpoints FAQ," *Software Engineering Journal,* vol. 11, No. 1, pp. 2-4, 1996.

[3] A. Baker *et al.*, "An experimental card game for teaching software engineering processes," *The Journal of Systems and Software,* vol. 75, No. 1-2, pp. 3-16, 2005.

[4] P. Senge, *The Fifth Discipline: The art and practice of the learning organization*, New York: Doubleday, 1992, 424 p.

[5] R. Gardner, *Games for Business and Economics*, New York: John Wiley & Sons, 1994, 496 p.

[6] C. Zapata, y G. Awad-Aubad, "Requirements Game: Teaching Software Project Management," *CLEI Electro-*

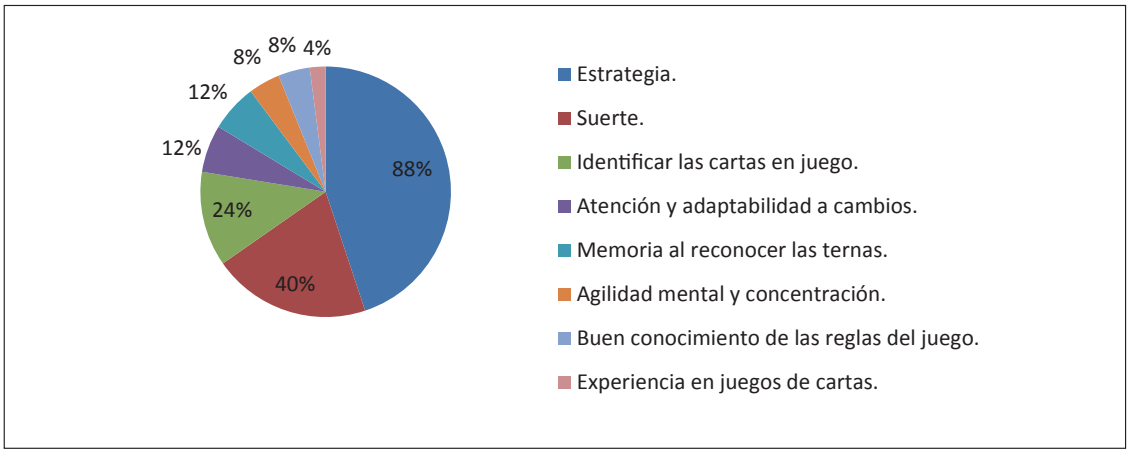

Figura 4. Categorías y porcentajes de los requisitos para ganar el juego

Fuente: elaboración propia.

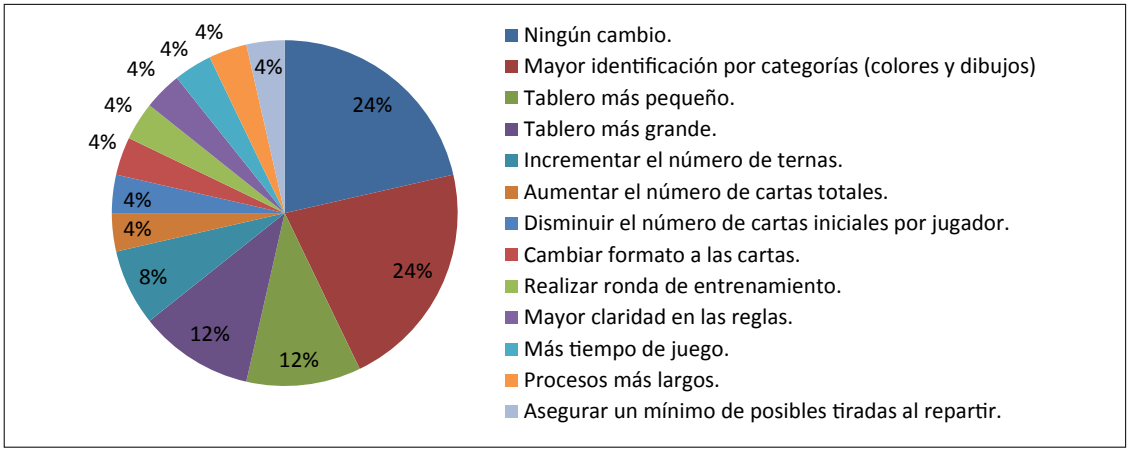

Figura 5. Categorías y porcentajes de las mejoras propuestas al juego

Fuente: elaboración propia.

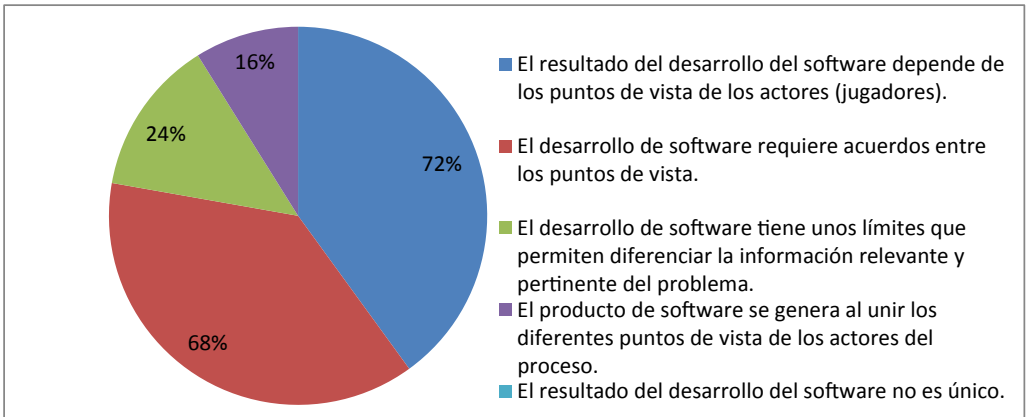

Figura 6. Cumplimiento de objetivos iniciales

Fuente: elaboración propia.

nic Journal, [En línea], vol. 10, *No.* 1, Disponible: http://www.clei.cl/cleiej/papers/v10i1p3.pdf, 2007.

[7] C. Zapata, y G. Giraldo, "El juego del diálogo de educción de requisitos," presentado en IV Congreso Colombiano de Computación, Bucaramanga, 2009.

[8] C. Zapata, y M. Duarte, "Consistency game: a didactic strategy for software engineering," *Rev. Téc. Ing. Univ. Zulia,* vol. 31, No. 1, pp. 3-12, 2008.

[9] C. Zapata, "Teaching Software development by means of a classroom game: the software development game," *Developments in Business Simulation and Experiential Learning,* vol. 36, pp. 156-164, 2009.

[10] G. Taran, "Using Games in Software Engineering Education to Teach Risk Management," presentado en Proceedings of the 20th Conference on Software Engineering Education & Training, Dublin, 2009.

[11] J. Leite, y P. Freeman, "Requirements Validation through Viewpoint Resolution," *IEEE Transactions on Software Engineering,* vol. 17, No. 12, pp. 1253-1269, 1991.

[12] A. Finkelstein *et al.*, "A Framework for Expressing the Relationships Between Multiple Views in Requirements Specification," *IEEE Transactions on Software Engineering,* vol. 20, No. 10, pp. 760-773, 1994.

[13] A. Abran *et al.* "Guide to the Software Engineering Body of Knowledge," [En línea], acceso julio 2011; Disponible: http://www.math.unipd.it/~tullio/IS-1/2007/Approfondimenti/SWEBOK.pdf, 2004.

[14] R. Fairley, "Educational Issues in Software Engineering," presentado en Proceedings of the 1978 annual conference, Washington, D. C., 1978.

[15] R. Fairley, "The Role of Academe in Software Engineering Education," presentado en Proceedings of the 1986 ACM fourteenth annual conference on Computer science, Cincinnati, OH, 1986.

[16] B. Boehm, "Software Engineering," *IEEE Transactions on Computers,* vol. C-25, No. 12, pp. 1226-1241, 1976.