THE EVENT INTERACTION GAME: UNDERSTANDING EVENTS IN THE SOFTWARE DEVELOPMENT CONTEXT

Carlos Mario Zapata
Universidad Nacional de Colombia
cmzapata@unal.edu.co

Paola Andrea Noreña Cardona
Tecnológico de Antioquia
panorena@tdea.edu.co

Fabio Alberto Vargas Agudelo
Tecnológico de Antioquia
fvargas@tdea.edu.co

ABSTRACT

The understanding of the context in which the software applications are developed is highly influenced by events. In fact, any organizational process is intended to be triggered by an event and sometimes is related to certain eventual-type results. Managing and modeling events is one of the most important tasks of the analyst during the software development process, since the behavior of the future software application is revealed by the event interaction. In the traditional way of teaching software engineering, events are commonly taught by means of lectures and "toy" practical projects. However, events are commonly misunderstood in this way of teaching, since they are often mistaken for operations. Also, the way in which an event interacts with each other is not completely understood by the students. Games are student-centered strategies for teaching software engineering concepts. Following a similar strategy to other software engineering games, we propose in this paper the event interaction game as a way to explain the meaning of the events and their interaction for understanding their influence on the software development process. This game is intended to surpass the problems generated by the traditional way of teaching events. The game was played by four heterogeneous groups of students belonging to different courses and universities, and the results are almost the same for such groups, leading to promising results in the understanding of events.

INTRODUCTION

Software engineering provides methods and techniques for analyzing the problems surrounding a software application. By understanding the context in which a software application should operate, we can develop the right application for solving such problems. Software engineering students experience some problems when they try to learn some topics about software development, because they should understand—by using some artifacts—the problem and the detailed structure of the solution before starting to write source code. Such understanding should be provided by the analyst. The final solution should be able to meet the stakeholder needs (Pressman, 2006; Claypool *et a*l., 2005; Sweedyk *et al.*, 2005; Barros *et al.*, 2008).

Software applications are usually developed in the interest of solving problems related to an organizational process. Such a process is commonly started by some situation—e.g. the arrival of a deadline, the completion of a previous process, etc.—called "triggered event." Also, a chain of processes often finish with another situation—e.g. the recording of some information, the calculation of some data, etc.—called "result event." To the extent of solving problems in software engineering, events have been taught by using lectures and "toy" practical projects, in which sometimes the students misunderstand events in some way—e.g. the interaction among events—and they confuse them with operations. Event representation is important in the software development context because functional requirements are commonly defined in terms of the event interaction (Weinbach & García, 2004), since events are responsible for changing the state of the processes affecting the sequence flow of activities and provide relevant information about the operation of the system, so they should be modeled in several artifacts (Hernández *et al.*, 2010).

Traditional teaching techniques include blackboards, oral lectures, books, and written exercises as the main vehicles to share knowledge. However, technological process brought new possibilities to teach and educate. Game technology is emerging for complementing classrooms, lectures, and activities with fun and interactive learning tools. Traditional methods are not enough to teach some concepts related to events, because such concepts are more easily understood in a practical way (Squire, Aguilera & Mendiz, 2003). Games can be effectively used for teaching several knowledge areas, and software engineering professors have explored this strategy for the teaching-learning process (Jain & Boehm, 2006; Birkhoelzer *et al,* 2005; Elbaum *et al.,* 2007; Nickel & Barnes, 2010; Eagle & Barnes, 2009).

In this paper, we propose a game for learning about the event interaction in the software development lifecycle by promoting the so-called event interaction graph. In favor of this proposal, we follow a similar strategy to other software engineering game designers. By playing this game, the students can clarify their event concepts and process. As a result, we promote this easy and funny way to learn about events, which is suitable for better understanding the process. This strategy can be used for teaching other software engineering issues, since it allows for surpassing the problems generated by the traditional way to teaching events. The game was played by four heterogeneous groups of students belonging to different courses and universities, and the results are almost the same for such groups, leading to promising results in the understanding of events.

The remainder of this paper is organized as follows: first, we present some ideas about the event interaction; next, we propose the event interaction game; then, we summarize the results of the game playing; finally, we conclude and establish the future work.

EVENT INTERACTION

The term "event" is used in the software development context as a synonym of *occurrence*. So, an event is a situation which occurs at a given time and place. The software applications are highly influenced by events during the development process, since usually the events start processes and affect the process status—influenced by the internal conditions of the system—and relationships (Oracle, 2005). Consequently, the early identification and analysis of events can be considered a good starting point for the requirements elicitation process (Singh *et al*., 2009). External events emerge at a certain time and designate the occurrence of something significant in a process, e.g. the beginning or the ending of an operation. In addition, they provide information about the behavior of a system for specifying the functional requirements (Weinbach & Garcia, 2004).

In the context of BPMN (Business Process Modeling Notation), an event is defined as something that happens in the course of business processes and affects the flows of such processes, generating either a cause or an impact (Owen & Raj, 2003). The term *event* is generally enough to

EXHIBIT 1
AN EXAMPLE OF EVENT INTERACTION GRAPH

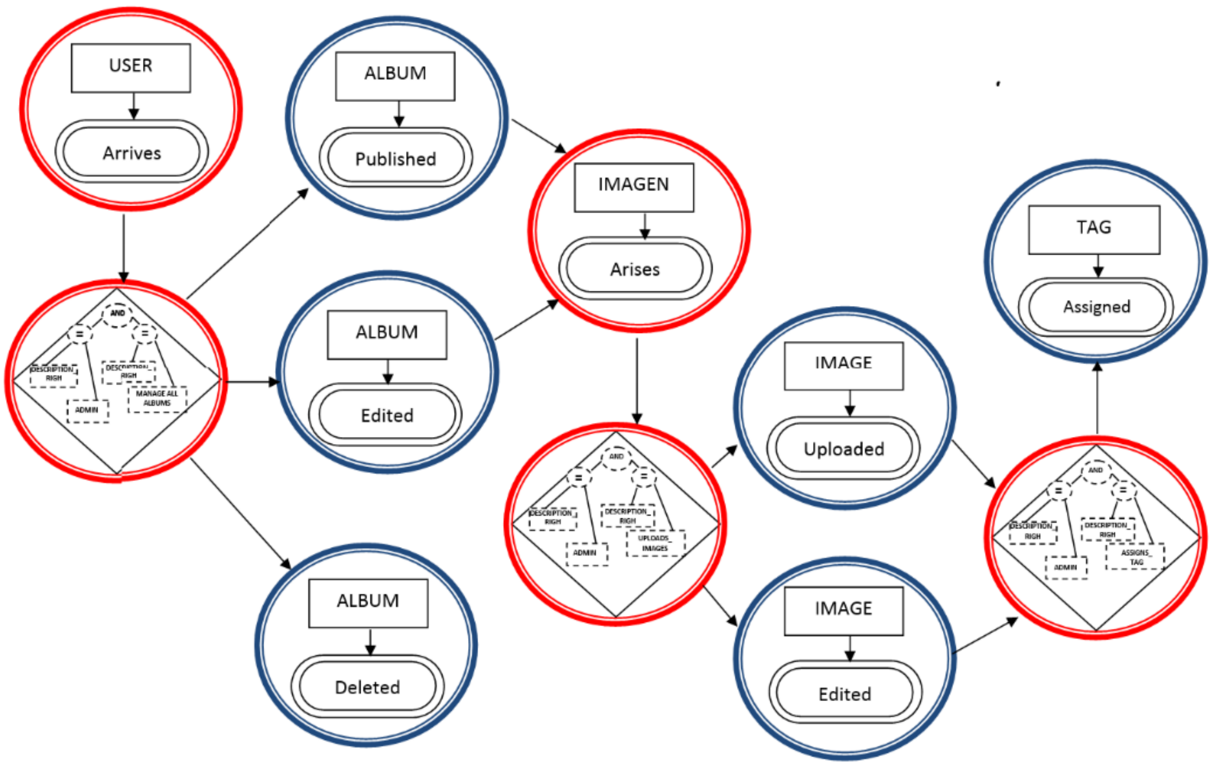
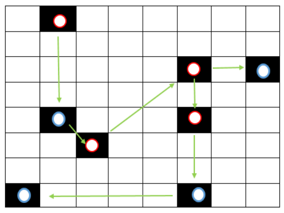


TABLE 1
GAME DESIGN

1. GENERALITIES	
Technique	Events Interaction Game
Game Objective	Learning about Event Interaction Graph, trigger events, and result events.
Players	2 to 4 teams. At least 2 players by team.
2. EDUCATIONAL COMPONENTS	
Topic	Event Interaction Graph
Purpose	Learning
Instructional Objectives	*Knowing the difference between trigger and result events *Understanding the interaction and sequence among events *Identifying the Event Interaction Graph by solving three boards
Basic concepts of the topic	Event, trigger events, result events, Event Interaction Graph, Sequence of events

3. TOOLS		
Name	**Quantity**	**Description**
Board of Event Sequence	12 (4 x team)	Every board has a 8x8 grid. Black squares are intended to locate events.
Nodes (Events)	128 (32 x team)	Board 1 = 5 nodes (3 usable) Board 2 = 8 nodes Board 3 = 8 nodes Graph = 11 nodes The blue nodes are result events. The red nodes are trigger events.
Chopsticks (edges)	60 (15 x team)	Every team will have 15 chopsticks for representing edges.

4. GAME RULES		
N°	**Description**	
1	The three boards are played by every team in subsequent rounds of the game.	
2	Every round a board and the nodes for the construction of the graph path are delivered to each team.	
3	Every team should locate the event nodes in the board black squares. The trajectory of the event sequence should be horizontal, vertical, and diagonal—45 degrees—among black squares of the board.	Example: 
4	The trajectory allows for: 2 trigger events together 1 trigger event followed by 1 result event 1 result event followed by 1 trigger event	It does not allow for: 2 result events together
5	The round ends when the first team has the correct location of the event sequence in the black squares of boards.	
6	Winners will accumulate points in each round: 5 points in the first board. 10 points in the second board. 10 points in the third board.	
7	When the 3rd round ends, every team should try the construction of the Event Interaction Graph. Every team have15 Chopsticks (they should use only 12) for the edges and 11 nodes for the events. The winner of the final round is awarded with 23 points.	
8	The sum of all these points will determine the winner of the game.	

encompass several moments along a business process: the starting of an activity, the ending of an activity, the change of a process state, and the arrival of a message, among others. In the Graphical User Interface (GUI) design, the term *event* is associated with functions and operations the user performs by clicking some hyperlink, choosing an option, or pressing a key (Osorio, 2008). Thus, in the software development lifecycle the term is often confused with event operations and functions. In the context of operations—or functions—an action is performed by an actor. In the context of events related to business processes, no actors are involved in performing the action, but a cause or a shot (Hernández *et al*., 2010). Therefore, if an event starts one or more processes is called *trigger event* (Oracle, 2005). Some examples of trigger events are: the emergence of a need, the arrival of a date, the ending of a month, the arrival of an order, etc. (Zapata & Arango, 2006). The trigger event is the starting point of a sequence of processes or other trigger events. A trigger event can be defined by using either a declaration, a restriction, or an action (Oracle, 2005).

What happens after the completion of one or more processes is called a result event. Result events are usually expressed in terms of verbs in past participle and can be linked to trigger events. For example, the result event *order filled in* can be attached to the trigger event *arrival of an order* (Zapata, 2012). In BPMN the result event is located at the end of the flow of a process (OMG, 2009). Managing and modeling the events is one of the most important tasks of the analyst during the understanding of the software context. The development of the right software application is highly influenced by the right elicitation of the events.

EVENT INTERACTION GRAPH

The Event Interaction Graph (EIG; see an example in the Exhibit 1) is a tool for representing the relationships and occurrence of events in a software application. Some authors commonly use this graph in the business logic for the sake of representing the event interaction within the system. The occurrences are synchronizations of the program units at any time and the relationships are interactions between two or more nodes in a sequence of events (Yuan & Memon, 2010). The EIG also represents all possible sequences of events to be executed in the GUI and the interaction between events and processes. The EIGs are, basically, discrete-time models with events in the nodes connected by directed arcs (arrows or edges); in this way an EIG can exhibit how events are linked to each other.

EVENT INTERACTION GAME

Event Interaction Game is intended to surpass the problems generated by the traditional way of teaching events by explaining the meaning of the events and their interaction. By using this game, we promote the understanding of the event influence on the software development process. Our game is divided into four boards: solution 1, solution 2, solution 3, and construction graph. All of the boards have events belonging to the same set, so the players can recognize the same events board by board. Also, the sequences of the three boards contribute to the construction of the graph at the end of the game. In the game, trigger events are colored in red, while result events are colored in blue. Game design is proposed in Table 1.

SOLUTION FOR EACH BOARD

1. Every team can try to solve the right sequence of events in each round. The board 1 allows for learning the technique of game for every board and it is supposed to be achieved in the minimum amount of time; boards 2 and 3 are more difficult and need more time. The sequence of all boards resembles every sequence of the final graph. Each graph solution is presented in Exhibit 2.
2. For the sake of clarity, we show the solution of the board 2 in Exhibit 3.
3. Exhibit 4 shows all nodes in the right positions.

EXHIBIT 2
RIGHT TRAJECTORY OF EVENTS IN THE GAME

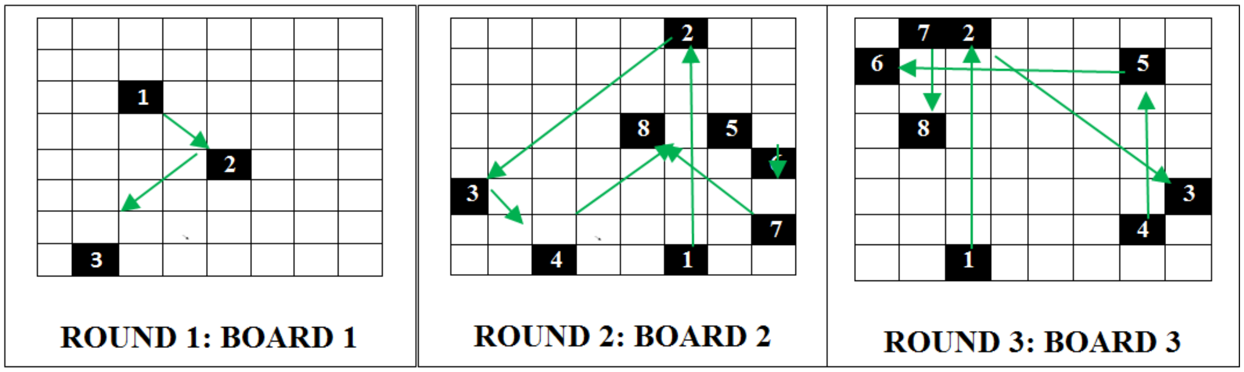


EXHIBIT 3
EVENT SEQUENCE IN THE BOARD 2

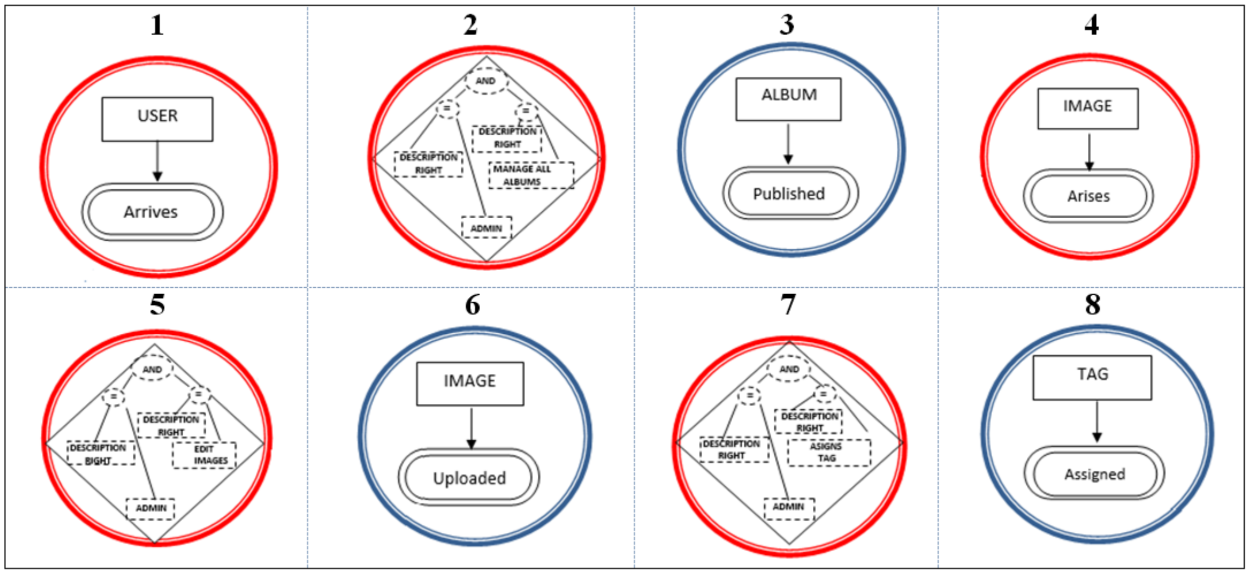
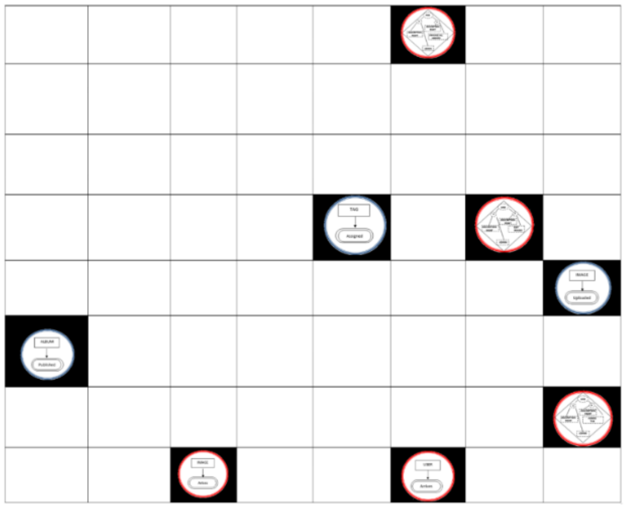


EXHIBIT 4
SOLUTION TO THE BOARD 2



4. Finally, the teams should construct the event interaction graph by recalling every sequence from boards (see Exhibit 1). 111)

RESULTS

We applied a survey to four heterogeneous groups of students belonging to three universities: Universidad Nacional de Colombia, Universidad de Medellin, and Institución Universitaria Técnologico de Antioquia. The game was practiced in 4 courses of Requirements and software Engineering by 85 students. We applied a 5-question survey in the interest of gathering some information from the players. The results are summarized in this Section.

1. **Did you understand the event functionality after the game?**

98% of the surveyed students answered *yes* (see Exhibit 5). This means the maximum of the students learned about the event functionality by practicing the game. The event interaction game was designed to give better understanding about events in an easy and funny way. The result shows that this strategy accomplishes the goal for which the game was created.

EXHIBIT 5
QUESTION 1

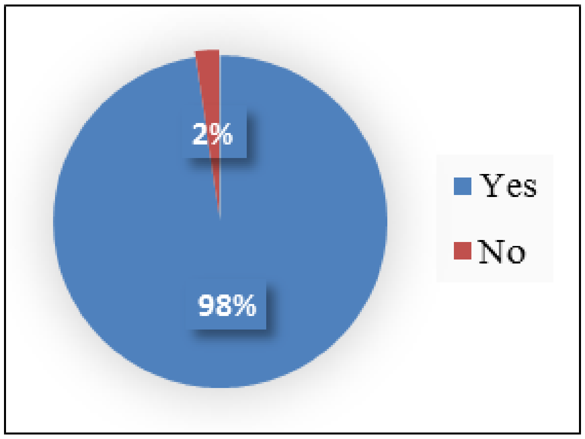
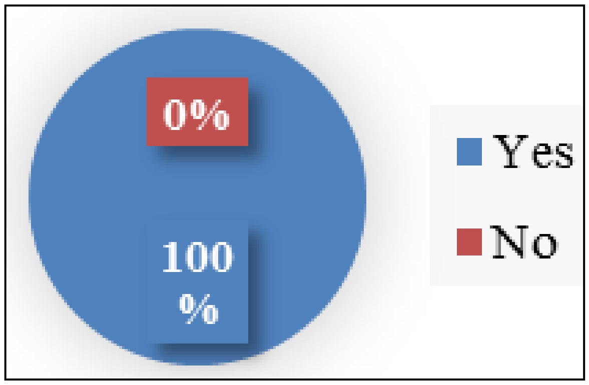


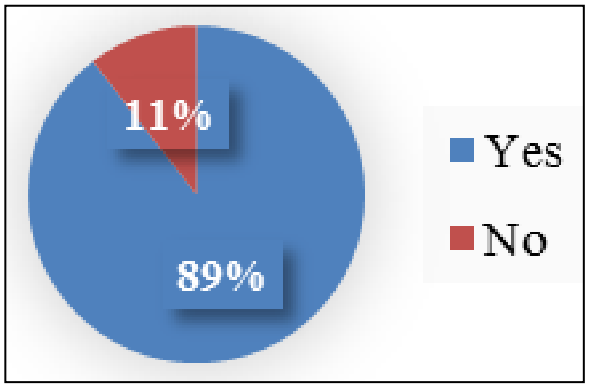
EXHIBIT 6
QUESTION 2



2. Did you understand the difference between the trigger events and the result events?

100% of the surveyed students answered *yes* (see Exhibit 6). This means the all students understood what are trigger events, result events, and the differences among them. Even though this is a difficult issue to discuss with the students, the game is promoting such an understanding by repeating once and again the nodes in the boards and the final graph.

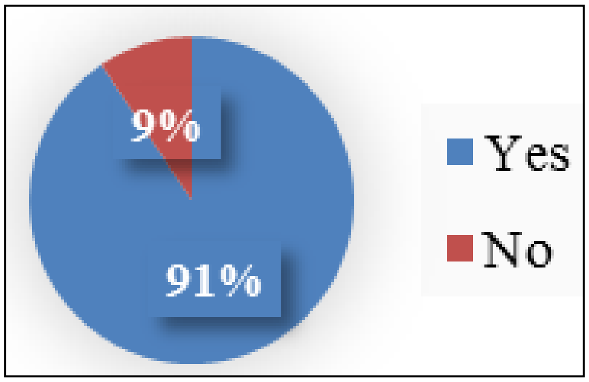
EXHIBIT 7
QUESTION 3



3. Did you learn about the Event Interaction graph after the game?

After the students discovered the meaning of event, trigger event, and result event by following the sequence of the game, they were able to create the event interaction graph. In This question, 89% of the surveyed students answered positively (see Exhibit 7), resembling the process of the game allows for learning about the usage and functionality of the event interaction graph.

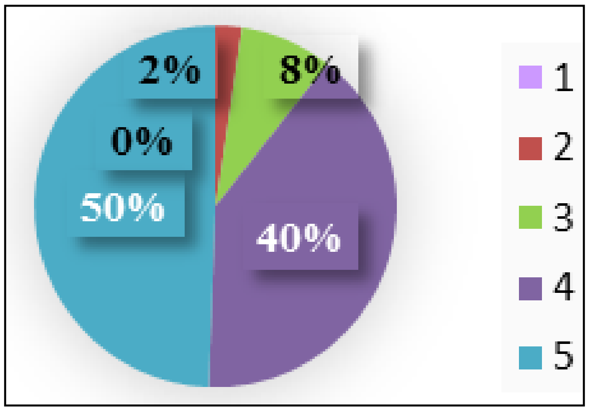
EXHIBIT 8
QUESTION 4



4. Do you believe this game eases the learning about events event interaction?

91% of the surveyed students learned about the event interaction (see Exhibit 8). In this case, the proposed learning strategy was useful. The usage of the same nodes for solving the boards reinforces the knowledge about events. Also, the "puzzle" they need to complete—i.e., the intended sequence of events—demands some discussion among students, promoting the possibility to learn from the game.

EXHIBIT 9
QUESTION 5.



5. Would you please provide a grade mark ranging from 1 to 5—1 means too bad, 2 means bad, 3 means fair, 4 means good, and 5 means Excellent—about the game?

90% of the students assigned a grade mark greater or equal to 4 (see Exhibit 9). 8% of the students consider the game *fair* and 2% consider it *bad*. The results are very promising, since the students were prone to participate in the game and they learned something from the game. Also, they considered the game funny and easy to play.

CONCLUSIONS AND FUTURE WORK

The game-based learning is a useful strategy for helping in the teaching-learning process. Such strategy proved to be effective for various levels of training and knowledge domains. By following this strategy, the event interaction game enables the learning process related to important elements of software engineering and business process. Event interaction game is an easy and funny way to learn about event interaction. Professors should be attracted to use this game, since it allows for team working in the classroom. We can infer from the 85 students surveyed in three universities that the proposed approach was accepted enthusiastically by participants, who recognize that the event interaction game is an effective way to learn important issues of software engineering and a way to explain the meaning of the events and their interaction. Students could also discover the influence of the events over the software development process. The results were satisfying and motivating.

Event interaction game leads us to several topics of future work:

1. Creating other scenarios for the event nodes.
2. Including more boards, since sometimes the game is ending in a fast way.
3. Creating a virtual version of the game, including some parameters for selecting the event nodes and the event sequences to be followed.
4. Defining some other issues to be taught by using games in the context of the software development process.

ACKNOWLEDGEMENT

We acknowledge Colciencias—the Colombian administrative department of Science, Technology and Innovation—for funding this project. The results of this paper belong to the Young Researcher project "A representation model for the events in the lifecycle software." We also acknowledge Nancy Estella Gonzalez Granados for her contributions in the initial stages of the game.

REFERENCES

Aguilera, M. & Mendiz, A. (2003). Video games and education: education in the face of a "parallel school") *Computers in Entertainment*, vol. 1, 1:1–1:10.

Arango, F. & Zapata. C.M. *UN-MÉTODO para la Elicitación de Requisitos de Software*. Universidad Nacional de Colombia: Medellín 2006.

Barros, M. O. & Araújo, Renata Mendes. (2008). Ensinando Construção de Software Aplicada a Sistemas de Informação do Mundo Real. In: I Fórum de Educação em Engenharia de Software, 2008, Campinas. *Anais do I Fórum de Educação em Engenharia de Software.*

Birkhoelzer, T., Navarro, E. Van der Hoek, A. (2005). Teaching by Modeling instead of by Models. *Proceedings of the 6th International Workshop on Software Process Simulation and Modeling*, St. Louis, MO. pp. 4.

Claypool, Kajal & Claypool, Mark. (2005).Teaching software engineering through game design. *ACM SIGCSE Bulletin*, 37, 3.

Eagle, M. & Barnes, T. (2009). Experimental evaluation of an educational game for improved learning in introductory computing," SIGCSE Bull., 321–325.

Elbaum, S. Person, S. Dokulil, J. & Jorde. M. (2007). Bug Hunt: Making Early Software Testing Lessons Engaging and Affordable," in Proceedings of the 29th international conference on Software Engineering, ICSE '07,(Washington, DC, USA), pp. 688–697, IEEE Computer Society.

Hernández, U., Álvarez, F.J & Vargas, M. (2010). Use Processes - Modeling Requirements Based on Elements of BPMN and UML Use Case Diagrams". *2nd International Conference on Software Technology and Engineering (ICSTE)*. 2010.

Jain, A. & Boehm. B. (2006). SimVBSE: Developing a Game for Value-Based Software Engineering. *Proceedings 19th Conference on Software Engineering Education and Training*, pp. 103 -114.

Neal, L. Predictions for 2003: E-Learning's Leading Lights Look Ahead. *E-Learn Magazine*, 1.

Nickel, A. & Barnes, T. (2010). Games for cs education: computer-supported collaborative learning and multiplayer games. Proceedings of the Fifth International Conference on the Foundations of Digital Games, FDG '10, (New York, NY, USA). 274–276.

OMG (Object Management Group). (2009). Business Process Model and Notation BPMN. *Standar Document.* http://www.omg.org/spec/BPMN/1.2. 2009.

Oracle. (2005) Documento número: B14220-02. *Documento estandár URL*: http://docs.oracle.com/cd/B19306_01/server.102/b14220/triggers.htm.

Osorio, F. (2008). *Lógica y programación orientada a los objetos: un inicio al desarrollo de software*. Medellín, Colombia: Instituto Tecnológico Metropolitano.

Owen, M. & Raj, J. (2003). BPMN and Business Process Management. Popkin Software.Technical report. http://www.omg.org/bpmn/Documents/6AD5D16960.BPMN_and_BPM.pdf

Pressman, R. (2006). *Software Engineering: A Practitioner's Approach*, 7ªedición, Mc Graw Hill.

Seas, K., & Brizee, A. (2010). APA style workshop. *The Purdue Online Writing Lab.* Retrieved from http://owl.english.purdue.edu/owl/section/2/10/.

Singer, L. & Schneider. K. (2012). It was a bit of a race: Gamification of version control. *Games and Software Engineering (GAS), 2012 2nd International Workshop*, 5 –8.

Singh, S.; Gupta, R. & SAbharwal, S. (2009). Automatic extraction of events from Textual Requirements specification. *Nature & Biologically Inspired Computing. NaBIC 2009*.World Congress, Coimbatore, India.415-420.

Squire, K. D. (2003). Video games in education. *International Journal of Intelligent Games & Simulation*, vol. 2, 49–62.

Sweedyk, Elizabeth Keller, Robert M. (2005). Fun and games: a new software engineering course. Proceedings of the 10th annual SIGCSE, 2005.

Weinbach, N. & García, A. Una Extensión de la Programación en Lógica que incluye Eventos y Comunicación. 2004.

Yuan X. & Memon A. Iterative execution-feedback model -directed GUI testing. *.Information and Software Technology*, 52, 559–575.

Zapata, C.M. (2012) *The UNC-Method revisited: elements of the new approach*. Saarbrucken: Germany.