A Game for Learning Event-Driven Architecture: Pre-conceptual-Schema-based Pedagogical Strategy

Paola Andrea Noreña Cardona
Universidad Nacional de Colombia
panorenac@unal.edu.co

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

ABSTRACT

An event is a piece of information for providing details about the state change of the processes of a software system and controls system behavior. Event-Driven Architecture (EDA) is a software architecture for promoting the production, detection, consumption, and reaction to events. Events in EDA trigger autonomous human or automated processing. EDA complements Service-Oriented Architecture (SOA) by employing events triggering services. EDA approach adds value to the enterprise by injecting value-added information. Some games are based on strategies for teaching system behavior by using processes, events and architectures. However, such games lack pedagogical strategies for teaching event functionality from EDA, which is necessary to the system behavior. We propose a game for teaching event functionality and the elements included in EDA by using a pre-conceptual schema (PS). Such schema is a computational modeling tool for representing a domain. PS includes structures for representing events, processes, and their relationships. We use such structures as pedagogical strategies, because the PS is a training and learning tool used in software engineering processes from academy and industry. The game is focused on teaching students and professionals in software system areas about functionality of events in EDA.

INTRODUCTION

An event is a piece of information about anything significant that happens. An event can affect processes (Edwards *et al*., 2011; Noreña *et al*., 2014; Zapata *et al.,* 2013). Information related to events has an event header and event body. Event header contains elements describing the event occurrence. Event body is used to describe what happened. The term *event* is used to refer the state change of the processes in a software system and it usually means a problem, an opportunity, a threshold, or a deviation of the system behavior (Michelson, 2006).

Event-Driven Architecture (EDA) is a software system architecture and a paradigm for event-based applications. Event processing is the central architectural concept (Luckham, 2002; Dunkel *et al*., 2011). So, in EDA, the system is divided into a set of so-called modules (Klusman *et al*., 2016). Each module may independently perform a particular task (Tragatschnig & Zdun, 2015). Among such modules, the communication is achieved by using the production, detection, consumption, and reaction to events. Some elements are used in such processes like: (i) event producer (or source); (ii) event consumer; (iii) event processing; (iv) scenario; (v) services; and (vi) events (Edwards *et al.,* 2011). Events in EDA happen inside or outside business and they are disseminated to consumer subscripted for triggering autonomous human or automated processing (Klusman *et al*., 2016).

Service-Oriented Architecture (SOA) is an architecture with all services defined. Such services may be used to perform simple functions or entire business processes. EDA is complemented by SOA because the occurrence of an event can trigger the invocation of one or many services. The interaction between SOA and EDA is a business intelligence approach. Such approach increases value to the enterprise as a neutral and independent manner of the hardware platform, the operating system, and the programming language in which the service is implemented (Michelson, 2006). So, SOA and EDA have received attention in academia and industry since they inject value-added information (Theorin *et al*., 2015).

Cagiltay *et al.* (2015), Chen (2014), Orojloo *et al.* (2017), Mulazzani *et al.* (2017), and Qin *et al.* (2016) present game approaches for teaching the system behavior by using their process. Some of them include events in their definition and graphical modeling. Anderson *et al.* (2017) present a game for defining events in behavior systems. Bartoletti *et al.* (2016) provide a game approach for defining event structures by using formal notation. Zapata *et al.* (2014) propose a game for understanding the meaning of the events and their interaction in the software development context. Herzig *et al.* (2012) and Matallaoui *et al.* (2015) expose approaches based on the creation of architectures for games including process and events. However, their pedagogical strategies are based on teaching processes, event definition and architecture development. So, the game approaches lack EDA elements, and pedagogical strategies for teaching event functionality from the architecture. Such functionality is needed for allowing knowledge system behavior from EDA, which is used for developing software products.

Accordingly, we propose Event-Driven Architecture Game (EDA Game) as an approach for teaching event functionality and the elements included in EDA by using a pre-conceptual schema (PS). EDA Game contains events in a pandemic scenario, *e.g*., infection rate increases, infection outbreak emerges; producers who should create events, *e.g.,* medic from a hospital; consumers who subscribe and receive notifications from such events by using event messages, *e.g.,* analyst from a laboratory. In order to win the EDA Game, the players should treat disease hotspots and researching cures for plague before another team win.

PS is a computational modeling tool for representing a domain. PS is close to the natural language by including unambiguous syntax and easing stakeholder understanding. PS combines graphic and linguistic structures in order to include structural and dynamic features of a software system *i.e*., nodes (concept, operator, conditional), links (connection, implication), relationships (structural, dynamic, and eventual), and gatherers (event, specification). Analyst can represent process, events and relationships in order to analyze the system behavior with such structures. PS-based events are used for triggering and dynamic relationships, which can be translated to either processes or services (Zapata, 2012).

So, we integrate PS structures to game as pedagogical strategy in four phases for understanding EDA: (i) selecting roles and tasks; (ii) knowing event functionality and processes related to events; (iii) attending services; and (iv) competing for the sake of reaching goals defined by the EDA Game. We select PS because they are tools used for training and learning software engineering processes from academy and industry. So, PS eases acquisition and retention of information and increment skills in order to achieve goals.

EDA Game is developed for teaching students and professionals of software system areas the theoretical concepts about EDA, their elements, and event functionality. The EDA Game experience is applied to student groups for validating the knowledge acquired during game.

This paper is organized as follows: in Section 2 we present the conceptual framework related to EDA; in Section 3 we present related work; in Section 4 we state the problem; in Section 5 we propose EDA Game for teaching EDA elements and event functionality; in Section 6 we apply such pedagogical strategy with students in software system areas. Finally, we summarize conclusions of the game and application, and establish some future work.

CONCEPTUAL FRAMEWORK

Event: piece of information emerging at a certain time and place (Edwards *et al*., 2011); events affect the processes and they are used for designating the occurrence of something significant that happens (Noreña, 2013; Zapata *et al*., 2013). Software systems are highly influenced by events during the development process, since an event provides relevant information about the behavior of the software systems. Such information is used in the functional requirements during the software development life cycle (Edwards *et al*., 2011; Noreña *et al.,* 2014; Zapata *et al*., 2013). Events can be a problem, an opportunity, a threshold, and a change of state in system behavior (Michelson, 2006).

Events contain an *event header* and an *event body*. *Event header* includes information about the events like id, event type, event name, and, specification. *Event body* describes what happened. Events should be used in computational products, *e.g*., process diagram, state machine diagram, activity diagram, elicitation requirements cards, etc. (Noreña, 2013). *Trigger* and *result* are event types. *Trigger event* is used for indicating the starting of processes and their occurrences. Such event can be timer (based on time), message (an alert), conditional (a restriction), and declaration or none (a sentence; OMG, 2009). Also, they can generate the execution of other trigger events. *Result event* is used for indicating the end of processes (Zapata, 2012).

Event-Driven Architecture (EDA): paradigm of software system architecture for easing high flexibility, scalability, and concurrency from event-based applications (Luckham, 2002; Dunkel *et al*., 2011). Communication between heterogeneous components is integrated in EDA. Such components perform a particular task by using the production, detection, consumption, and reaction to events (Klusman *et al.,* 2016; Tragatschnig & Zdun, 2015).

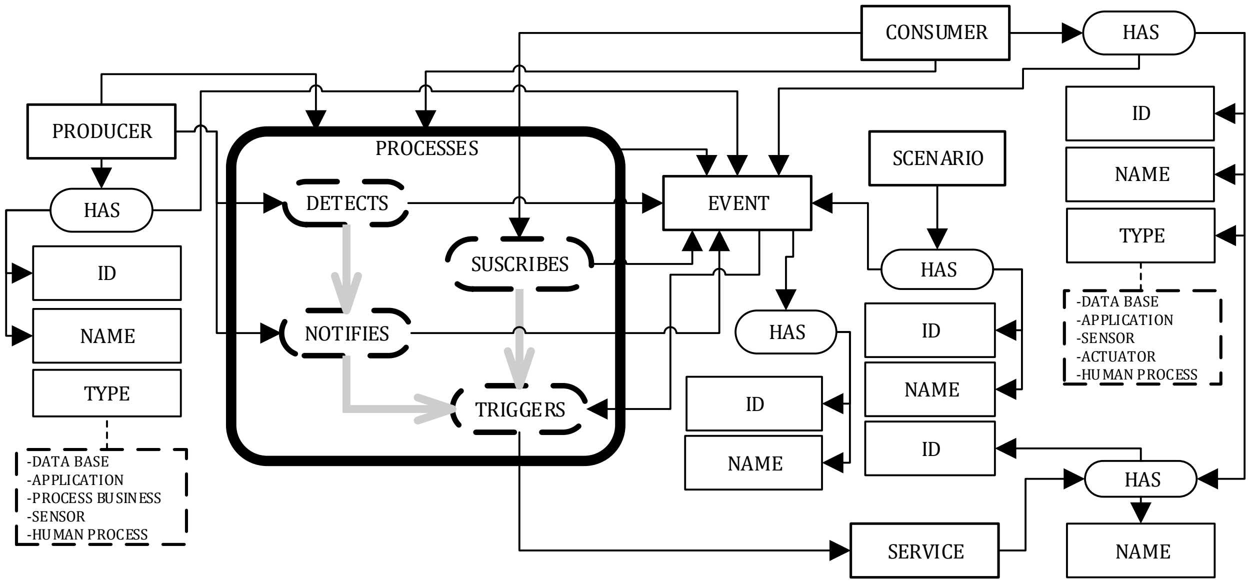
EDA contains some elements with the following description—see the pre-conceptual schema base on Edwards *et al*. (2011) in Exhibit 1—*producer* detects and notifies an *event; producer type* can be data base, application, business process, sensor, and human process; *consumer* subscribes *event; consumer type* can be data base, application, business process, actuator, and human process; producer and consumer *process* an event, such processing is represented in the PS as a frame including the dynamic relationships producer *detects* and *notifies* event, consumer *subscribes* event, and event *triggers* service; *scenario* is a domain; *service* is an action or process triggered by an event; event *triggers* a service and can be timer, message (alert), conditional, and a declaration or none.

Service-Oriented Architecture (SOA): service architecture defined by using a description language and interfaces in a network (Maréchaux, 2006). A business intelligence approach is the interaction between SOA and EDA. Such approach adds exponential value to the enterprise by injecting value-added information (Michelson, 2006; Theorin *et al*., 2015). EDA and SOA are combined for easy integration among business units and heterogeneous platforms and environments (Maréchaux, 2006). So, both architectures receive attention in academia and industry (Michelson, 2006).

Pre-conceptual Schema (PS): A modeling tool close to natural language for graphically and computationally representing the domain of a problem and translating it to conceptual schemas and source code. Analysts and stakeholders can easily understand

and validate the information described in the PS. Consistence among structural and dynamic features of a software system is guaranteed in PS by using the following structures (see Exhibit 1): *nodes* can be concepts (nouns and noun phrases, *e.g*., producer, consumer, name); *relationships* can be structural relationships (verbs generating permanent connections between concepts, *e.g.,* consumer has name), dynamic relationships (processes or services, *e.g.,* producer detects event), and eventual relationships (verbs in events, *e.g.*, emerges, passes showed in Exhibit 3); *links* can be implications (arrows of cause and effect, *e.g.,* if detected, modifies) and connections (arrows between concepts, *e.g.,* event triggers service); *gatherers* can be events (which triggers and ends dynamic relationships *e.g.,* message (alert) emerges showed in Exhibit 3) and frames (integrates other elements *e.g.,* the dynamic relationships: producer detects and notifies event; Zapata, 2012).

EXHIBIT 1.
PS ABOUT EDA ELEMENTS



RELATED WORK

Some authors present approaches for teaching processes. Such games are based on a set of processes, but such approach lacks pedagogical strategies for teaching event functionality and elements from EDA. Cagiltay *et al.* (2015) develop a game for teaching processes by using entity-relationship diagrams for modeling data in relational databases, but excluding event functionality. Chen (2014) presents a game for teaching process sequences by using state transition diagram. However, the game lacks pedagogical strategies for teaching event functionality and elements from EDA. Orojloo *et al.* (2017) propose a game-theoretic approach for modeling security of systems by using a state graph including events, but such approach is only used for teaching processes. Mulazzani *et al.* (2017) present a directed acyclic graph as a Bayesian network of alternative models for understanding processes, but excluding event functionality. Qin *et al.* (2016) expose a game from probabilistic machine learning model for representing processes integrating events. However, such a game lacks pedagogical strategies for teaching event functionality and elements from EDA.

Other authors present approaches for teaching event definition. Anderson *et al.* (2017) propose a game for controlling event-driven behavior. Such game allows for recognizing and responding to different events occurring in the game environment, but excluding teaching elements from EDA. Bartoletti *et al*. (2016) provide a game from a formal definition of event structures with some conditions and services. However, such a game lacks pedagogical strategies for teaching event functionality and elements from EDA. Zapata *et al.* (2014) present an event interaction game for understanding events in the software development context. However, such a game is oriented to explain the meaning of the events and their interaction, but such a game lacks pedagogical strategies for teaching event functionality and elements from EDA.

Finally, some authors present approaches for teaching event functionality from the architecture. Matallaoui *et al.* (2015) develop a model-driven serious game development by using the Gamification Modeling Language (GaML) and Herzig *et al.* (2012) expose a generic platform for enterprise gamification. However, their pedagogical strategies lack event functionality and elements from EDA.

PROBLEM STATEMENT

Academia and industry in software system areas lack a game with pedagogical strategies for teaching students and professionals the event functionality and the elements from EDA. Such a problem is observed because the game approaches are generally used for explaining the behavior of software systems from processes and some include events, but their pedagogical strategies are focused on teaching processes; other game approaches are focused on defining events and event functionality from the architecture. Also, the games use some diagrams, but excluding elements from EDA and pedagogical strategies for teaching event functionality and elements from EDA. Event functionality and elements from EDA are needed for allowing knowledge system the behavior of software systems from EDA; So, students and professionals can use such architecture for developing software products.

EVENT DRIVEN ARCHITECTURE (EDA) GAME

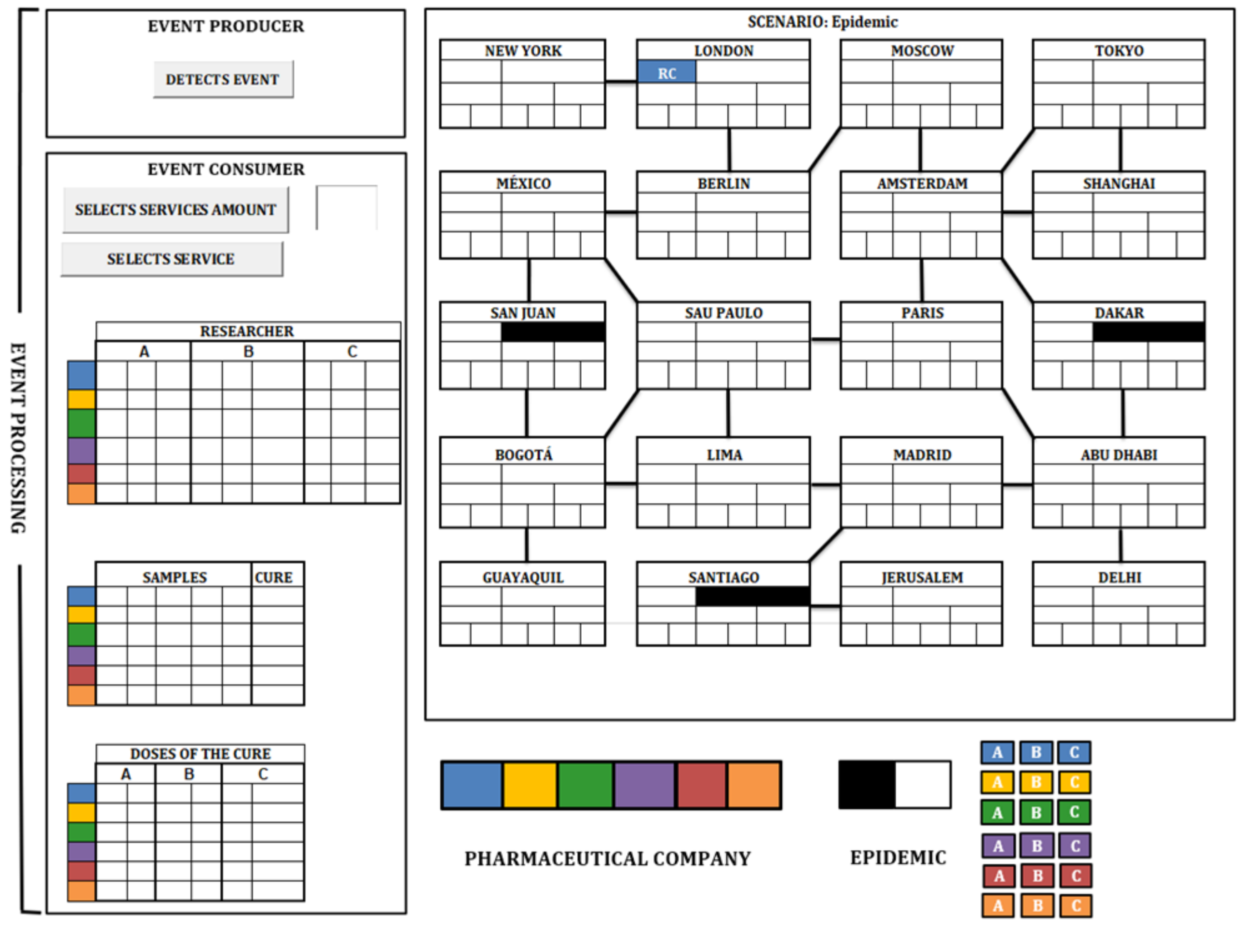
Pedagogical Strategy

Game Objective: learning about event functionality and elements from EDA. EDA Game is developed in order to instruct students and professionals in software system areas theoretical concepts about EDA.

Phases: (i) selecting roles and tasks, companies have resources for fulfilling the mission; (ii) knowing event functionality and services related to events, infection can arrive to city; (iii) attending services, companies can use services for obtaining more resources; and (iv) competing for the sake of reaching goals defined by the EDA Game, companies should cure epidemic infections in most cities for winning.

Tools: EDA Game platform can be seen by all players (see Exhibit 2). Such platform contains the following elements: scenario determines the place when are detected and notified events i.e., world cities, where can be built research centers, can be produced epidemic infections, and can be cured cities and researchers; event producer detects and notifies event, and an event will emerge; event consumer selects services amount and services according to the amount generated. In consumer zone the number of researcher infections is included, which can be three; the epidemic samples quantity of every company, the cure, and doses of the cure of researchers, event producer and consumer zones integrate the event processing; event triggers service i.e., time passes, epidemic arrives to city (with research center), epidemic arrives to city (without research center); such events are displayed in a window from the platform (see Exhibit 3), the event epidemic arrives to city is presented with event message alert emerges for notifying the event occurrence; service should be used by researchers i.e., travel to city, build research center, explore epidemic sample, discover epidemic cure, treat epidemic city and treat researcher; such services are displayed in other window from the

EXHIBIT 2
EDA GAME PLATFORM



platform (see Exhibit 4).

Events and services are represented by using PS structures *i.e.,* eventual relationship with one or two concepts in a circle for events and dynamic relationship for services. We use such structures as pedagogical strategy because PS a is software engineering tool for training and learning processes for developing software systems from academia and industry. So, professional and student analysts acquire and retain information by using PS and increment skills in order to achieve goals from the game.

Game Instructions

Initial conditions: (i) selecting from two to six teams, such teams will be pharmaceutical companies; (ii) each team should select a color of the pharmaceutical company; (iii) each team should select a city without epidemic for building their first research center; (iv) each team will have three researchers for investigating and curing the epidemic.

Mission: Company should cure most cities.

Event processing: (i) event producer detects event; (ii) researcher notifies event and an event will emerge: *epidemic arrives to city (with research center),* should occur in one city with research center away to the position of a pharmaceutical research center by using event processing; event information in software systems is notified by using event *message (alert) emerges* (see Exhibit 3.a.); *epidemic arrives to city (without research center)* should occur in one city with research center away to the position of a pharmaceutical research center by using event processing; event information in software systems is notified by using event *message*

EXHIBIT 3
EDA GAME EVENTS IN PS NOTATION

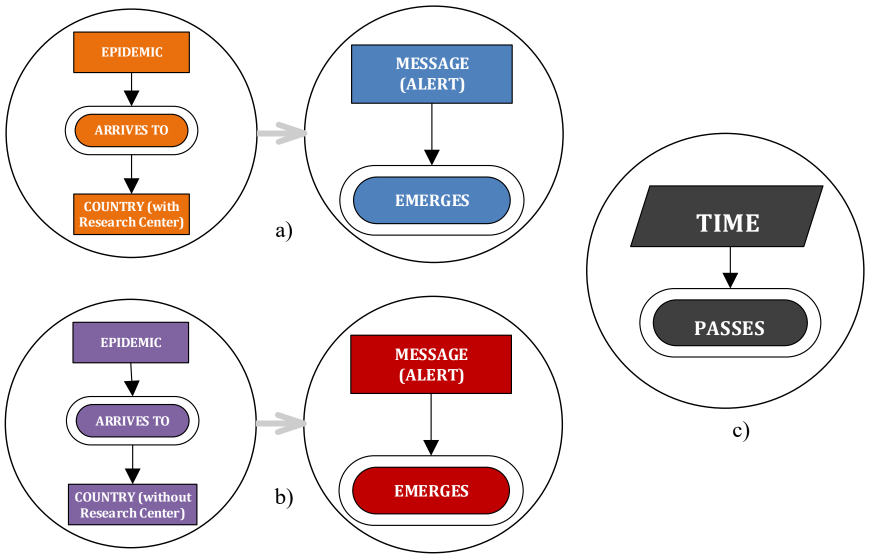
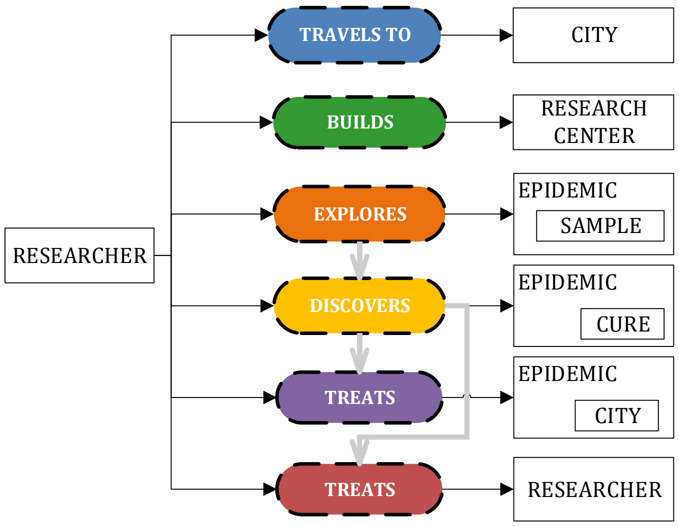


EXHIBIT 4
EDA GAME SERVICES IN PS NOTATION



(alert) emerges (see Exhibit 3.b.); *time passes* triggers services with no infection of a city (see Exhibit 3.c.); iii) events trigger *services*, *event consumer* selects *services amount* and select *services* according to the amount generated. Also, the services are used by the researchers from pharmaceutical company and they should be always different. The company should say the services they will use before using them.

Services are: *travel to city* is mandatory as a first service, in each event processing should be used, in order to travel to another city, one researcher should pass throughout two connected cities without repeating the road; *build research center* can be used when all researchers from pharmaceutical company arrive to the same city; *explore epidemic sample* can be used when a researcher arrives to an infected city and then returns to the nearest research; *discover epidemic cure* can be used after collecting five epidemic samples, a researcher should go to his research center and take two doses of the cure; *treat epidemic city* can be used after finding epidemic cure, in an event processing can be used one to two doses of cure by a researcher; *treat researcher* can be used after finding epidemic cure, one to two doses of cure can be used by a researcher in an event processing; whether a researcher is in a city at his research center and has three epidemic infections cannot move from place. Then, another researcher should go to such city for curing him (see Exhibit 4).

APPLICATION OF THE PEDAGOGICAL STRATEGY

The EDA Game experience is applied to two groups for validating the knowledge acquired during games: group 1, 25 students of requirements engineering in the system engineering program at Universidad Nacional de Colombia and group 2, 15 practitioners in a software company from Medellin, Colombia. Validation is achieved in two phases (i) poll before game and (ii) poll after game. In the *poll before game* we included two questions: Do you know you about EDA? What is event functionality? in which, both groups knew event functionality and services, but in other architectures and they do not knew EDA. The results in the *poll after game* are explained in Exhibit 5.

EXHIBIT 5
RESULTS OF EDA GAME APPLICATION

1. Defining 3 elements from EDA		
Answers	*Group 1*	*Group 2*
3 right elements	14	13
2 right elements	9	2
1 right elements	2	0
2. What is event functionality from EDA?		
Answers	*Group 1*	*Group 2*
Events trigger services	9	4
Events produce services	4	0
Events enable services	7	5
Events change system behavior	3	1
Events change system states	2	1
Events call services	0	1
Events start services	0	3
3. What is fun factor of EDA game? (from 1 to 5)		
Fun factor	*Group 1*	*Group 2*
5	12	11
4	8	2
3	3	2
2	2	0
1	0	0

We should observe from *poll before game,* pedagogical strategy by using PS structures improves knowledge about event functionality and elements EDA. Most students from group 1 and practitioners from group 2 achieved to describe elements from EDA as follows: events, producer, consumer, scenario, services, and event processing; and the most remembered elements were events and services for their graphical modeling in pre-conceptual schemas. Such answers indicated pedagogical strategy used in EDA Game allowed for learning about elements from EDA. Also, all students from group 1 and practitioners from group 2 defined correctly event functionality from EDA as follows: events trigger, produce, call, and enable services; and events change states and system behavior. Such answers demonstrated the ease of learning event functionality from EDA by using EDA Game. During the game we analyzed how the different teams had a good time and enjoyed playing EDA Game *e.g.,* when epidemic infected cities and researchers, and when researchers cured cities and other researchers. So, the most students from group 1 and practitioners from group 2 evaluated fun factor of EDA Game with the highest score (5). The fun factor is a main feature of the games for teaching scientific and technologic knowledge (Vázquez & Manassero, 2017).

CONCLUSIONS

Games improve traditional pedagogical strategies for teaching students and professionals in software system areas, complementing the practices and the theory activities.

EDA Game is a funny game, which was focused on teaching students and professionals in software system areas about event functionality and elements from EDA. Such event functionality and elements are used for developing software products; so, such game was needed for allowing knowledge system behavior from EDA.

PS structures were used as pedagogical strategy in EDA Game, because the PS allows for training and learning in software engineering processes from academy and industry.

Experiential learning about event functionality and elements from EDA by using EDA Game improved knowledge professionals in software system areas.

We suggest as future work the application of the EDA Game in professional environments for analyzing reactions and knowledge among participants. Also, other games can be created for teaching other topics in software engineering.

AKNOWLEDGMENTS

This paper is product of the Ph.D. research project *an extension to pre-conceptual schemas for refining event representation and mathematical notation* with Hermes code 39886 at Universidad Nacional de Colombia. This work is sponsored by Colciencias (Administrative Department of Science, Technology and Innovation) from Colombia in the program *Ph.D. students in Colombia,* bid 727 of 2015.

REFERENCES

Anderson, P. R., Friedman, R. M., Gagner, M. B., Gronkowski, T. T., Michael, J. I. I., Shi, V. T., & Walsh, J. L. (2017). Controlling event-driven behavior of wagering game objects. U.S. Patent No. 9,542,807. Washington, DC: U.S. Patent and Trademark Office. https://www.google.com/patents/US20110021263

Bartoletti, M., Cimoli, T., Pinna, G. M., & Zunino, R. (2016). Contracts as games on event structures. Journal of Logical and Algebraic Methods in Programming, 85 (3), 399-424. http://www.sciencedirect.com/science/article/pii/S235222081500036X

Cagiltay, N. E., Ozcelik, E., & Ozcelik, N. S. (2015). The effect of competition on learning in games. Computers & Education, 87, 35-41. http://www.sciencedirect.com/science/article/pii/S0360131515001001

Chen, Z. H. (2014). Exploring students' behaviors in a competition-driven educational game. Computers in Human Behavior, 35, 68-74. http://www.sciencedirect.com/science/article/pii/S0747563214000818

Dunkel, J., Fernández, A., Ortiz, R., & Ossowski, S. (2011). Event-driven architecture for decision support in traffic management systems. Expert Systems with Applications, 38(6), 6530-6539. http://www.sciencedirect.com/science/article/pii/S0957417410013254

Herzig, P., Ameling, M., & Schill, A. (2012, August). A Generic Platform for Enterprise Gamification. In Software Architecture (WICSA) and European Conference on Software Architecture (ECSA), 2012 Joint Working IEEE/IFIP Conference on. IEEE.

Klusman, M., Plasmeijer, R., & Wolter, R. (2016). Event-Driven Architecture in software development projects. Master Thesis Computing Science, Radboud University, Nijmegen.

Edwards, M., Etzion, O., Ibrahim, M. Iyer, S. Lalanne, H., Monze, M., Moxey, C., Peters, M., Rabinovich, Y., & Sharon, G. (2011). Un modelo conceptual para los sistemas de procesamiento de eventos. IBM Technical document. https://www.ibm.com/developerworks/ssa/webservices/library/ws-eventprocessing/index.html

Luckham, D. (2002). The power of events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Boston: Addison-Wesley.

Qin, Z., Khawar, F., & Wan, T. (2016). Collective game behavior learning with probabilistic graphical models. Neurocomputing, 194, 74-86.

Maréchaux, J.L. (2006). Combining Service-Oriented Architecture and Event-Driven Architecture using an Enterprise Service Bus. IBM Technical document.

Matallaoui, A., Herzig, P., & Zarnekow, R. (2015, January). Model-Driven Serious Game Development Integration of the Gamification Modeling Language GaML with unity. In System Sciences (HICSS), 2015 48th Hawaii International Conference on. IEEE.

Michelson, B. M. (2006). Event-Driven Architecture overview. IBM, Patricia Seybold Group Technical document, 2. http://www.cioindex.com/nm/articlefiles/66181-EventDrivenArchitectureSOA.pdf

Mulazzani, L., Manrique, R., & Malorgio, G. (2017). The Role of Strategic Behaviour in Ecosystem Service Modelling: Integrating Bayesian Networks with Game Theory. Ecological Economics, 141, 234-244.

Noreña, P.A. (2013). Un mecanismo de consistencia en los eventos disparador y de resultado para los artefactos de UNC-Method. Tesis de maestría. Universidad Nacional de Colombia, Medellín, Colombia.

Noreña, P.A, Vargas, F. & Soto, D. (2014). Tipificación de eventos a partir del modelo BPMN en artefactos de ingeniería de software. Cuaderno Activa, 6, pp 49-61. http://ojs.tdea.edu.co/index.php/cuadernoactiva/article/view/200

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

Orojloo, H., & Azgomi, M. A. (2017). A game-theoretic approach to model and quantify the security of cyber-physical systems. Computers in Industry, 88, 44-57.

Theorin, A., Bengtsson, K., Provost, J., Lieder, M., Johnsson, C., Lundholm, T., & Lennartson, B. (2015). An event-driven manufacturing information system architecture. IFAC-PapersOnLine, 48(3), 547-554.

Tragatschnig, S., & Zdun, U. (2015, June). Modeling Change Patterns for Impact and Conflict Analysis in Event-Driven Architectures. In Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2015 IEEE 24th International Conference on. IEEE.

Tran, H. & Zdun, U. (2013). Event Actors Based Approach for Supporting Analysis and Verification of Event-Driven Architectures. 17th IEEE International Enterprise Distributed Object Computing Conference. Vancouver, BC.

Vázquez-Alonso, Á., & Manassero-Mas, M. A. (2017). Juegos para enseñar la naturaleza del conocimiento científico y tecnológico. *Educar*, 53(1), 149-170.

Zapata, C.M. (2012). The UNC-Method revisited: elements of the new approach. Eliciting software requirements in a complete, consistent, and correct way. Saarbrucken: Lambert.

Zapata, C. M., Noreña, P. A., & González, N. E. (2013). Representación de eventos disparadores y de resultado en el grafo de interacción de eventos. Ingenierías USBmed, 4(2), 23-32. http://revistas.usb.edu.co/index.php/IngUSBmed/article/view/288

Zapata, C. M., Noreña, P. A., & Vargas, F. A. (2014). The Event Interaction Game: Understanding Events in the Software Development Context. Developments in Business Simulation and Experiential Learning, 41. https://journals.tdl.org/absel/index.php/absel/article/view/2123