Adv Syst Sci Appl 2021; **04**; 1-15
Published online at https://ijassa.ipu.ru.

Simulating Events in Requirements Engineering by Using Pre-conceptual-Schema-based Components from Scientific Software Domain Representation

Paola A. Noreña C.[1,2*], Carlos M. Zapata J.[1]

[1)] *Facultad de Minas, Universidad Nacional de Colombia, Medellín, Colombia*

E-mail: paolanorena@itm.edu.co

[2)] *Facultad de Ingenierías, Instituto Tecnológico Metropolitano, Medellín, Colombia*

E-mail: cmzapata@unal.edu.co

Abstract: Event simulation is a process for analyzing the behavior of a natural or automated system. Such behavior is produced by events–phenomena or something that happens at any moment. Events are simulated in scientific software domains (SSD) for finding scientific results and developing scientific software, which allow making critical decisions. Usually, scientists use scientific models with simulation components for simulating events and requirements engineering (RE) models for representing SSD in a software development process. However, the separation of simulation components in scientific models and software components in RE models causes inconsistency in elements of the system domain, especially in events. Consistency is required for a near solution to the needs of the scientific stakeholders. Business analysts also use RE models and know software development processes, but they lack scientific knowledge for describing elements of a SSD. Therefore, scientists and business analysts require a computing model, which integrates common knowledge by using scientific and software components for simulating events and producing scientific software. Pre-conceptual schemas (PCS) are computing models used on RE for representing a domain, which is near to stakeholders. Several simulation components are found in PCS, i.e., events. Thus, we propose PCS components for simulating events in RE from SSD. After, we validate the PCS understand and usage in SSD. Scientists and business analysts can take advantage of the PCS for representing and simulating events and developing scientific software from the same model. Such a proposal contributes to reducing the gap between both engineering and science into scientific software development.

Keywords: event simulation, pre-conceptual schemas, requirements engineering, scientific software domains, scientific software

1. INTRODUCTION

Event simulation is a process used for studying the behavior of several complex systems by using events [21]. Such a simulation decreases cost and risk for analyzing natural or automated systems [6]. Events are instantaneous occurrences or phenomena of such systems [15]. Events are responsible of system behavior by changing the state of processes and triggering other events in a time sequence, e.g., volcano erupts, patient suffers heart attack, signal decreases, and earthquake arrives [20]. Commonly, a scientific software is developed in scientific software domains (SSD) for making critical decisions and solving real-life problems by using event simulation, e.g., making weather predictions based on climate models [10]. Examples of scientific software domains are mathematics, engineering, medicine, and sciences [11–13]. Consequently, the event simulation presents new challenge to scientific software development processes.

[*] Corresponding author: panorenac@unal.edu.co

Scientific software is characterized by including a complex knowledge based on event simulation. Rus et al. [22], Brosig et al. [2], González et al. [7], González et al. [8], Bonaventura and Castro [1], Song et al. [23], Camejo et al. [3], Laurindo et al. [12] use scientific simulation models for describing the complex knowledge of a SSD and analyzing alterations and impacts of the system from events. Simulation component set of a model comprise clock, initial state, variables, parameters, events, activities, attributes, event generator, and random variable generator [6]. Such authors also use requirements engineering (RE) models with software components for representing the domain in a software development process. Software component set of a RE model comprise structural (entities and relationships) and dynamic (the system flow, processes, and events) features for representing a domain.

González et al. [7] recognize the growing need for simulation and programming complex systems. But both simulation and software components are separated into two or more scientific and RE models, generating inconsistency in the element representation of the scientific software domains. Since several components in different notations cause some elements may disappear or be confused with other elements in the development process, interfering with the understanding of the domain. In addition, scientists create scientific applications without a RE process and business analysts lack scientific knowledge, taking them more time for understanding a SSD [9]. Thus, scientists and business analysts require a computing model, which allows integrating a common and consistent knowledge with elements of simulation and software for simulating events in SSD from the same model.

Pre-conceptual schemas (PCS) are conceptual models based on object-oriented computing used in requirements engineering for representing a domain containing events. The PCS are models near to natural language of stakeholders for easily understanding the complete view of software systems by using dynamic and structural features [25]. The notation in PCS involves scientific and simulation components for representing events and mathematical equations, i.e., initial conditions, variables, events, parameters, etc. [20], which are used in event representation of SSD as: electronics [16], chemistry [20], environmental engineering [5], geology [18], industry [19], epidemiology [26] and oil engineering [24].

Consequently, in this paper we propose simulation and software components integrated into PCS for simulating events in requirements engineering of a SSD. RE is used for understanding problems of stakeholders and solving them by applying software engineering methods previously to the software development phase, which can be used in SSD [13]. After, we validate the PCS understanding and usage in scientific software domains with scientists and engineers. Such a solution is proposed for business analysts and scientists can use the PCS as a computing model integrating common knowledge from software and science fields by using software and simulation components for recognizing, analyzing, designing, simulating, and coding the functionality of involved events to SSD as an advance to reducing the gap between both fields. This integration between simulation and software components offers a greater understanding of real-world events in a domain and a greater consistency among the modeling, simulation, and implementation phases.

This paper is structured as follows: in Section 2 we define the conceptual framework; in Section 3 we state the problem; in Section 4 we propose a solution; in Section 5 we validate the proposal. Finally, we discuss conclusions and future work in Section 6.

2. CONCEPTUAL FRAMEWORK

2.1. Events

Occurrences that happen in the dynamic view of complex systems [14]. Events are important elements because their information is used in the functional requirements during the software development life cycle for controlling the system behavior. Also, events are used for changing the state of processes [15]. Such changes occur by accomplishing constraints, which are conditions for processing events, activities, and data [20]. A trigger is an event used for initiating processes and other events in the system in time sequence; such a sequence is used for controlling the execution of event flow in the system [17].

Trigger event types are *none* or *statement* (instruction for triggering a process, e.g., earthquake arrives); *conditional* (constraint for specifying a requirement, e.g., if seismometer alert = "active"); and *timer* (time for controlling events and processes, e.g., the statement event "time passes" with a cycle for increasing seconds from *0* to *28800 seconds* [20]).

2.2. Event simulation

Process used for analyzing structural and functional behavior of different complex systems, which are controlled by using events. Also, event simulation is used for studying alterations and impacts of a system by keeping the real system intact [6]. Events are simulated by using models, which are used for representing the system by using mathematical, logical, structural, and dynamic structures. Such a simulation is applied for predicting the effects of changes in the system by using events and significantly reducing their risks and costs [7]. Components are the elements and modules used from event simulation.

Elements. Clock (counter for saving changes in the system, which are produced by events in discrete times for the beginning and the end of a simulation); *initial state* (set of variables and parameters for describing the system state); *entity* (concepts for describing roles or resources of the system, e.g., patient, student, room); *attribute* (concept for describing features of an entity, e.g., name, age); *variable* (number or concept for representing a variable value in a simulation process, e.g., door is opened); *parameters* (constant value, e.g., pi = 3.1416); *event* (occurrence for changing states of the system, e.g., volcano erupts); and *activity* (concept for representing an action e.g., engineer measures water lever) [6, 21].

Modules. Initialization (module for starting the system by using the initial state); *clock* generation (module for updating the clock simulation); *random variable generation* (module for creating the system behavior); and *event generation* (module for changing the state system) [6, 21].

2.3. Pre-conceptual Schemas (PCS)

Conceptual model used in requirements engineering for computationally representing and recognizing main elements of a domain as events in the software engineering process. PCS link structural and dynamic features for describing the complete view of a software system [25]. Also, elements with linguistic, mathematical, and graphical structures are included in the PCS components for representing events and mathematical notation [16, 20] (see Fig. 1).

Components. Structural relationship (verb for relating dependence among classes or entities and leaf concepts or attributes), *dynamic relationship* (verb for representing processes or activities), and *eventual relationship* (verb for representing events); *concept* (node for representing class and leaf concepts); *specification* (gatherer for assigning values or integrating operations), *constraint* (gatherer for operations with conditions and derived values), and *frame* (gatherer for including reports or operations, which are same types); implication (connection for linking dynamic relationship and event flow), *connection* (connection for linking concepts and relationships), *concept-note* (connection for linking specification, constraint, and value-note; [25]), and *operation* (connection for linking operator, concept, variable, vector, and parameter); *value-note* (gatherer for representing value of a variable, parameter, and leaf concept), *class-leaf concept* (node for summarizing structural relationship among class and leaf concepts); *operator* (node for representing mathematical equations), i.e., + (sum), sin (sine), sqr (square root); *timer* (event for controlling time of other events and processes [17]); *event* (occurrence for triggering dynamic relationships and other events, which can be have a specification or a constraint); vector (node for saving information of the system); *variable* (node for representing a variable value); parameter (node for representing a constant); and *initial conditions* (gatherer for starting variables and parameters of the system [20].

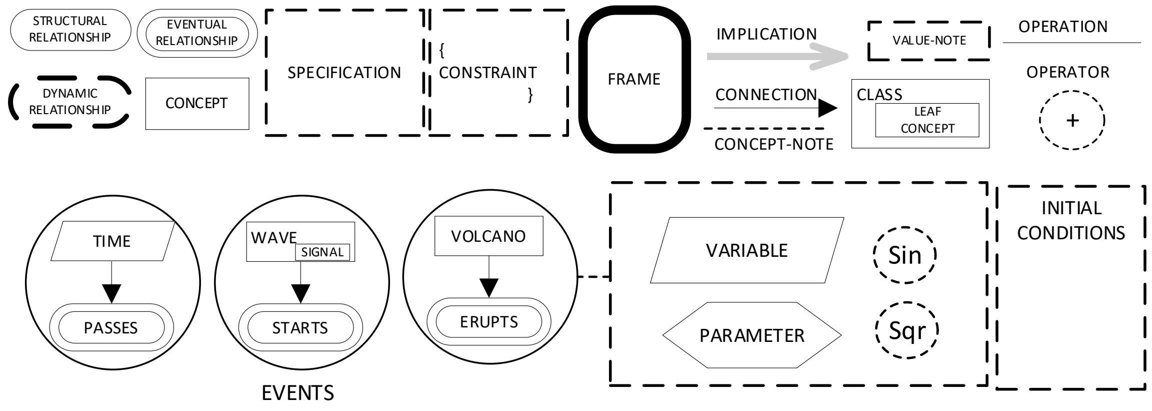

Fig. 1. PCS components based on [20]

2.4. Scientific Software Domain (SSD)

SSD are focused on solving the needs of scientific studies and research by using scientific software systems. Such systems are used as tools for understanding phenomena, which can be analyzed from real systems avoiding risks and impacts, e.g., biologic, chemical, physical, and environmental systems. Such software systems play an important role in critical decision making and solution of real-life problems in fields as mathematics, engineering (e.g., electronic, electrical, mechanic, etc.) medicine, and sciences (e.g., geography, chemistry, meteorology, geology, etc.) [11, 13]. According to Kanewala and Bieman [10], scientific software types are industrial (a system developed for complex processes), research (a system developed for specific research in a scientific context), and small-scale (a system developed for specific solutions of science students).

3. PROBLEM STATEMENT

Rus et al. [22] represent four models from RE as a process model, cause-effect diagram, UML class diagram, and a simulation model (a discrete event model) for arriving at the implementation phase, the information of the several models is integrated into the simulation model. Although they check consistency among such diagrams, the models have different elements. Brosig et al. [2] use UML diagrams for representing RE with parameters, services and actors, and Petri nets for representing activities and transitions in a simulation process.

Discrete Event System Specification (DEVS) is the most general formalism for modeling systems with discrete events. However, scientists have little experience in requirements and software project practices [1,9]. González et al. [7] use DEVS for executing models using simulation protocols, they recognize growing need for simulation and programming complex systems. So, they use UML state machine diagrams with simulation elements (events, activities, constrains and states) in an automatic teller machine (ATM) case. However, the diagrams lack a description logic of the system since it is inconsistent with the code. González et al. [8] indicate the modeling and simulation for design and prototyping of systems are widely used techniques, they agree in the usage the Model Driven Development (MDD). Therefore, the authors use UML state machine diagrams for describing structural and dynamic features of a software system with an object-oriented paradigm with simulation elements as: events, activities, states, and transitions, but such a representation lacks several elements for a complete representation and simulation of events in the same ATM case. Bonaventura and Castro [1] propose a methodology for complex software projects based on Modeling and Simulation (M&S) formal, DEVS, and software project practices in a scientific project about questions of the origin of the world. The methodology is a very close solution, but such a project is represented in a high level of data networks. Then, the simulation only is applied to the network and both the methodology, and the project lack requirements specifications.

Song et al. [23] propose a grey box testing method for availability simulation software based on the event tree model. This model allows event simulating and modeling with elements as: events, parameters, initial conditions, logic and sequence events, and relationships among elements. But the graphical representation applied in the software testing phase, only includes nodes with numbers for the events and edges for the relationships, logic, and sequence, which. Camejo et al. [3] propose a computational simulation model for milk production process, which is developed in the programming language *Arena*, a simulation software based on events and processes. So, they use elements as entities, attributes, activities, events, and mathematical models. However, the model is used by a flow diagram with general processes. Laurindo et al. [12] present a communication mechanism between an event simulation software and a dynamic software, using a and a conceptual model (a graph with events, processes, and time) and a similar simulation model in the software simulation. Nevertheless, the integration lacks other elements of domain necessary in the requirement engineering.

In Summary, the authors use separately two or more kinds of models with several notations for integrating event simulation and software components related to scientific software development process: (i) a conceptual model from requirements engineering (e.g., UML diagrams and other models from complex systems) including elements as: entities, attributes, events, and activities from SSD (ii) a simulation model with elements as: time, events, activities, initial conditions, parameters, variables, and complex notation in mathematical models. Such separation between event simulation and software components generates inconsistency in the modeling, simulation, and implementation in a scientific software development process as: some elements may disappear or be confused with other elements in the development process causing interferences in the understanding of the domain. (iii) Scientists usually create scientific applications without a requirement engineering process and business analysts lack scientific knowledge, taking them more time for understanding the SSD [9]. Consequently, a computing model is required, which allows integrating a common and consistent knowledge with components of simulation and software for simulating events from RE process in SSD.

The PCS notation integrates (i) software components from RE and computational linguistics as structural and dynamic features for representing the complete view of any domain [25]; (ii) scientific components as linguistic, graphical, and mathematical structures for representing events are included from SSD, which can be used in the modules for simulating events [20], (iii) PCS are used in event representation of signals [16], chemical mixtures [20], noise environmental [5], geology [18], industry [19], epidemiology [26] fluid pressure of the petroleum [24], and simulation [20] for developing scientific software. So, we select the PCS as a computing model for proposing a solution.

4. PROPOSAL SOLUTION

4.1. Identifying PCS components (elements) for simulating events from SSD

We observe the functionality of components in event simulation, and we compare it to the functions of PCS components (see Fig. 1) for identifying common elements; then, we find the elements of the Table 1, which have same function. Such elements also are related to mathematical notation into the model.

Table 1. PCS elements of event simulation based on [6, 20]

Simulation component	*PCS (software) component*	*Function*
Initial state	Initial conditions	Integrating initial variables and parameters
Clock	Timer	Controlling the simulation time
Entity	Class concept	Relating main concepts, which have leaf concept
Attribute	Leaf concept	Featuring concepts of the classes
Event	Event	Triggering dynamic relationships events, and state changes of the system
Activity	Dynamic relationship	Relating processes to the system
Variable	Variable	Saving variable values or data
Parameter	Parameter	Saving a constant value or data

4.2. Defining PCS components (modules) for simulating events from SSD

Initialization. Module for automatically starting the system behavior by using the initial conditions of the PCS. Such conditions contain initial values of parameters and variables for simulating events, e.g., the parameters simulation time = 28800 seconds, threshold = 0,7, and rock rigidity = 7 x 107 Kg/meters^2 and the variables timestamp = "next", time = 0 seconds, random value = 0,0, tectonic plate state = "position", wave acceleration = 0,0 meters/seconds2, seismometer alert = "inactive", and seismograph state = "off"; which are initial values for analyzing the behavior system about an earthquake.

Clock generation. Module for automatically updating the simulation time and controlling other modules by using a timer of the PCS (see Fig. 1). Such an event is used for generating the beginning and end of a simulation by increasing time variable (a sum); we can use second-minute-hour units and digital format of time. Timer types are cyclical time (when the event has a cycle for increasing the time, e.g., time passes has a cycle from 0 to 28800 seconds) and specific date (when the time is a parameter used in a condition, [19]). Also, we include timestamps, which are variables used for saving time moments when events happen; thus, they can start or stop the time [14–15], e.g., timestamp = "stop" when a seismometer alert = "active".

Random variable generation. Module for automatically changing system behavior of the PCS. Random variables are used in simulation and statistical probability for giving a value and analyzing their effects. Random function *rand ()* is integrated to the PCS for generating random values between 0 to 1 (random variable = rand, see Fig. 2a), random values between an upper limit and a lower limit (random variable = *rand ()* * upper limit + lower limit, see Fig. 2b), and a random variable between two values (random variable = *rand ()* linking two values, see Fig. 2c); however, if another function for a random variable is required can be represented by using the PCS notation (see Fig. 1).

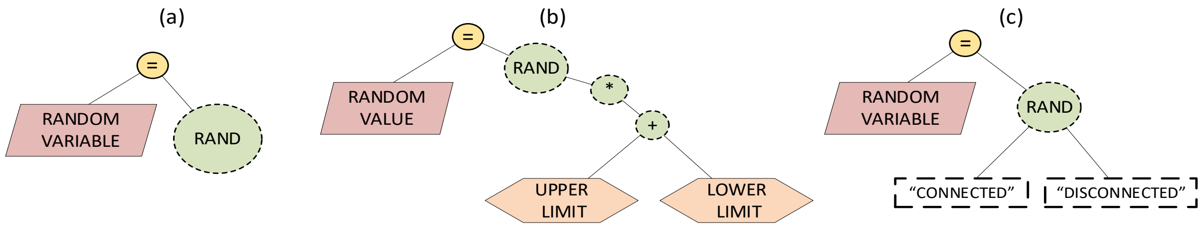

Fig. 2. Random variable generation

Event generation: Module for automatically operating and changing states of variables of the system by using event sequence in the PCS. Such a sequence allows for generating autonomous processes by using initialization, clock generation, and random variable generation modules. Then, events are executed from their internal function for changing the system behavior by using their specifications and constraints, which include dynamic relationships for inserting and updating data by accomplishing conditions of time values, states of variables, and initial values, e.g., earthquake arrives, seismometer alert emerges, and station measurement starts and ends.

4.3. Simulating events in requirements engineering from SSD by using PCS

We propose the PCS usage steps for simulating events in requirements engineering from SSD, which are exampled by representing an early earthquake alert generation system as a lab study in the SSD of Geology. Scientists use such systems for simulating events, predicting, and avoiding risks of damages and disasters.

Representing elements of SSD. Scientists and business analysts should represent the elements of the system domain by using PCS components (elements) as classes with its attributes (leaf concepts), e.g., *seismologist* (with *code* and *name*), *tectonic plate* (with *code* and *name*), *earth zone* (with *code, name*, and *tectonic plate*), *station* (with *code, name, location, latitude, longitude,* and *altitude*), *measurement* (with *local time, slip distance*, and *fault area*), and *earthquake* (with *date moment time, focus, epicenter, latitude, longitude, final slip, distance, final fault, area, seismic moment, magnitude,* and *intensity*) are classes of the system identified and represented in the Fig. 3.

Representing the flow of SSD. Scientists and business analysts should describe the complete logic and flow the system. Such description will be translated to a programming language in the development phase of the scientific software. First, they should use the PCS components (modules) as *initialization, clock generation*, and *random variable generation* (if other variables or ranges are required, then the *variable* component should be used). Then, they should include PCS components (elements) as initial conditions, timer, events, dynamic relatioships (processes or activities), variables, and parameters. Values, processes, and events can be detailed by using especifications, which include mathematical equations or internal conditions. Finally, the logic and flow of processes and events is related by using the *implication* link if a implication is required (from an event or condition to a process, from a process to other process, and an event to other event).

Example, the *initialization* module is represented in initial conditions for starting parameters: *simulation time, threshold,* and *rock rigidity* and variables: *simulation time, time random value, tectonic plate state, wave acceleration, seismometer alert,* and *seismograph* *state* (See Fig. 3.) The *clock generation* module is represented in Fig. 3. with the *time passes* timer, which start the system flow by using a cycle from *0 seconds* to the *simulation time* (*28800 seconds* according to the initial conditions) if *timestamp = "next" and time <= simulation time*.

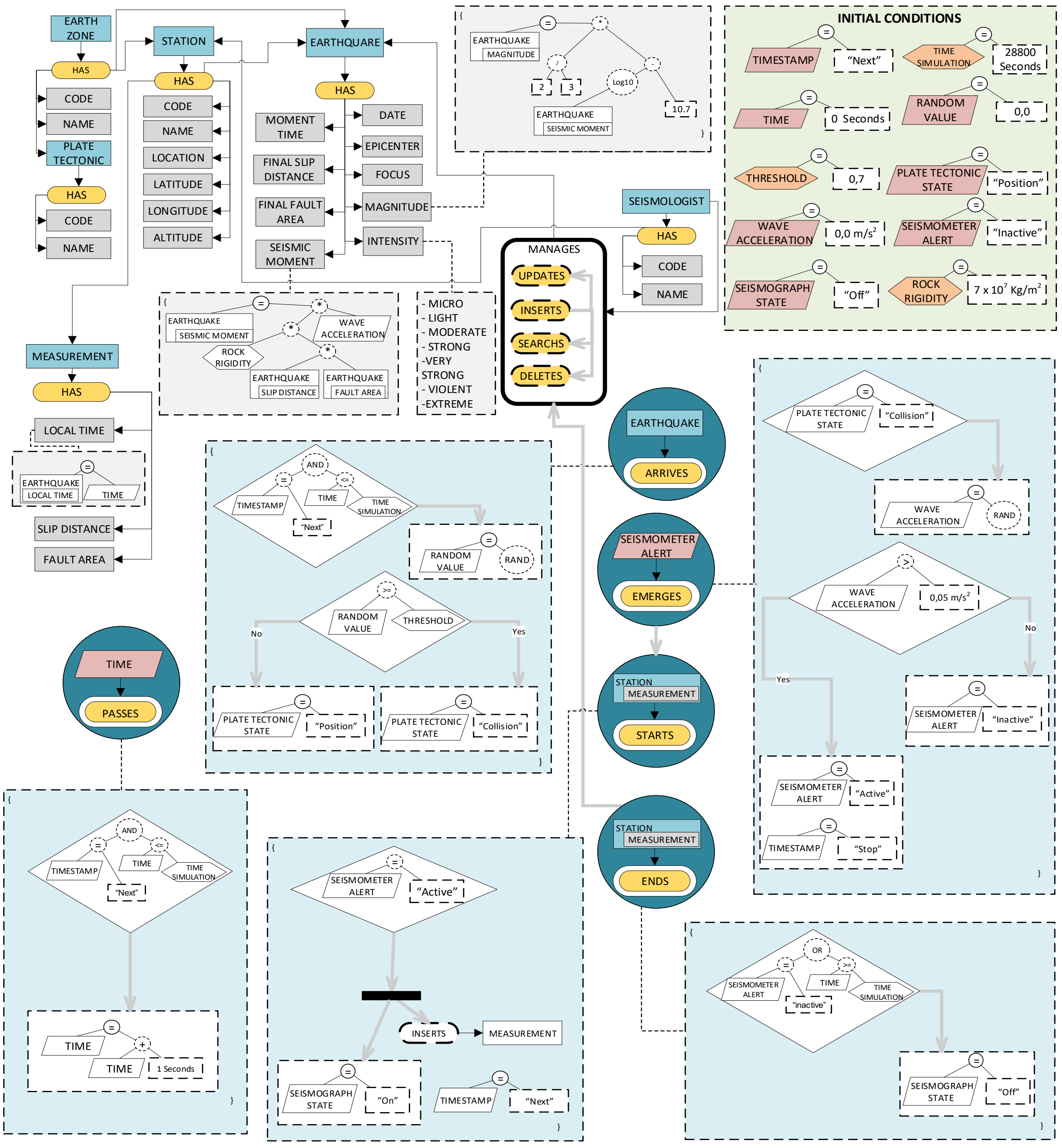

Fig. 3. Simulation event model by using PCS

Earthquake arrives event is random event that happens only when a random value fulfils the condition If *random value >= threshold* (threshold = 0,7 according to the initial conditions); when a tectonic plate collides, another random variable *wave acceleration* in *seismometer alert* event is generated (*random variable generation* and *event generation* modules are applied). If *wave acceleration > 0,05* m/s^2 (minimum value of risk) then *seismometer alert* is activated, also the time is stopped by using *timestamp = "stop"* else *seismometer alert = "inactive."* Such an event triggers the *measurement station starts* event when *seismometer alert = "active"* for changing the *seismograph state* to *"on"* and automatically inserting the measurement data (*local time, slip distance, fault area*); in this case, we use the implication link for obligatorily relating to the cause of other event. Thus, *timestamp = "next"* for increasing the time and returning to *seismometer alert emerges*, comparing the wave acceleration value, and saving measurement to the end of the earthquake. If *seismometer alert = "inactive" or time >= simulation time*, then *station measurement ends*.

When measurement ends, then *seismologist manages—updates, inserts, searches, deletes,* which are dynamic relationships—the *earthquake* class. Such a class has leaf concepts *time moment, date, epicenter, focus, final slip distance, final fault area, magnitude, intensity,* and *seismic moment. Magnitude* and *seismic moment* are derived attributes (values calculated by applying mathematical equations with leaf concepts inserted, numbers, and parameters; *magnitude = 2/3 * (log 10 seismic moment – 10,7)* and *seismic moment = rock rigidity * slip distance * fault area * wave acceleration*; the *intensity* is *micro, light, moderate, strong, very strong, violent*, and *extreme* according to the magnitude.)

Simulating events from SSD by using PCS. The complete logic allow performing the event simulation with data required in the system and the database. Thus, both scientists and business analysts should validate the requirements with scientific stakeholders with representation, then they can understand the complete system, simulating events and its data previously to the development phase of the scientific software.

Example, we simulate events in an earthquake occurred in Santander, Colombia. Such a phenomenon is analyzed by using the PCS of the Fig. 3. We show the epicenter and station in Fig. 4, where the earthquake was measured and detected. We use the real information of variables, parameters, classes, and leaf concepts for simulating the earthquake.

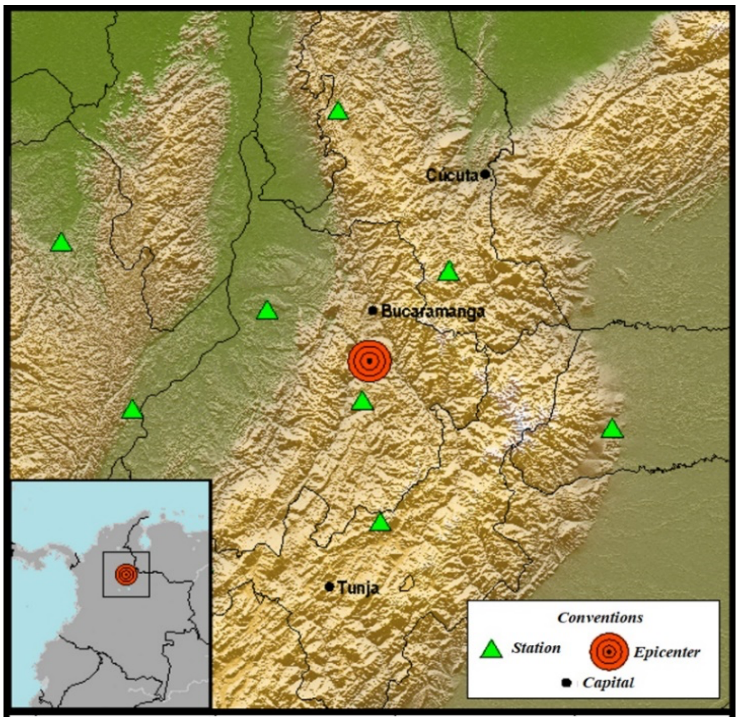

Fig. 3. Earthquake occurred in Santander, Colombia [4]

Values (leaf concepts) of the *tectonic plate, earth zone, seismologist,* and *station* classes are stored in the tables of the database *tectonic plate, earth zone, seismologist*, and *station* (see Tables 2, 3, and 4). Values (*local time, slip distance, fault area,* and *station code*) of the measurement class are automatically inserted by using the event *station measurement starts* (see Fig. 3) and stored in the table measurement of the database (see Table 6; *wave acceleration random* value assigned is '1,2 m/s^2' for applying the mathematical equation of the seismic moment, its unit is Newton (N) meters. Values (*date, moment time, earth zone code, station, focus, epicenter, latitude, longitude, final slip distance, final fault area, seismic moment, magnitude,* and *intensity)* of the *earthquake* class are inserted by the *seismologist* when the *station measurement ends* event happens (see Fig. 3) and stored in the table *earthquake* of the database (see Table 7). Earthquake had an *intensity* "strong" and a *magnitude* '5,5 N meters', value summarized in '5,5' by excluding units in Richter scale. Some fictional information is added to the real data in order to complete the simulation, e.g., the *seismologist*.

Table 2. Tectonic Plate. The Authors

Code	*Name*	*Earth Zone Code*
PTC1	Cocos	101
PTC2	South American	101
PTC3	Nazca	101
PTC4	Caribbean	101

Table 3. Earth zone The Authors

Code	*Name*
101	Colombia

Table 4. Earth zone The Authors

Code	*Name*
104521452	Juan Manuel Gaviria

Table 5. Station. The Authors

Code	*Name*	*Location*	*Latitude*	*Longitude*	*Altitude*	*Earth zone code*	*Seismologist code*
62	Barranca	Santander	7,101°	-73,7°	132	101	104521451

Table 6. Measurement. The Authors

Local time (s)	*Slip distance (m)*	*Fault area (m^2)*	*Station code*
30000	0,5	33	62
31000	1,5	34,1	62
32000	2	36,3	62
33000	2,8	38,5	62
34000	3	39	62
35000	3,35	40,2	62

Table 7. Earthquake. The Authors

Date	*Moment Time(s)*	*Earth Zone Code*	*Station code*	*Focus (m)*	*Epicenter*	*Latitude*	*Longitude*	*Final slip Distance (m)*	*Final fault Area (m^2)*	*Seismic Moment (m)*	*Magnitude*	*Intensity*
30/05/2018	3000	101	62	140.000	Santander	6,83°	-73,11°	3,35	40,2	94269x 105	5,5	Strong

5. VALIDATION

We evaluate the understanding and the usage the PCS components by the expert opinion. We collect the data with a survey as instrument. We select two independent variables: current role and area. The sample size is 28 experts (10 scientists: 6 in Geology, 1 in Energy, 1 in Petroleum, and 2 in Simulation; 11 developers, 3 professor, and 4 analysts in RE and software engineering) and four dependent variables:

PCS representation. First, experts observed the PCS in Fig.3. After, they answered the question *Q1. Describe with your words, what is being represented in the model?* for analyzing the PCS representation variable. We observe 61% of experts described phrases related to an earthquake early warning system, simulation, detection, and automated measure of an earthquake, i.e., a system to measure earthquakes and tectonic activity, tectonic plate movement simulation, a system to detect and collect information on earthquakes. In addition, the 28% of experts indicated phrases related to an earthquake, a tectonic activity, a seismic event, and events of an earthquake and a seismological station, i.e., a seismologist takes measurements of an earthquake, a conceptual model of earthquakes, their properties, and measurements. Only 11% of experts described different phrases as: relational model, a system, and remembers. The results indicate the 89% (25) of experts perform a near description to the system domain (See Fig. 4).

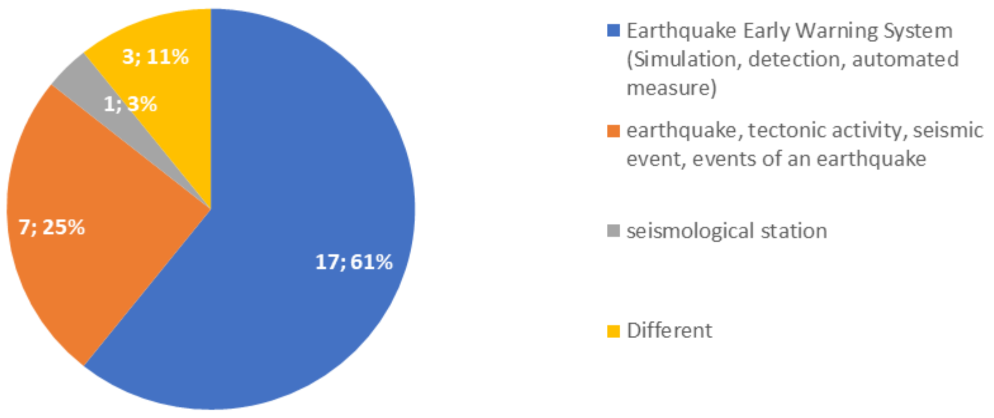

Fig. 4. PCS representation

Component integration. Experts answered the question: *Q2. Does the PCS integrate scientific, software, and simulation components?* for analyzing of the component integration variable. The 86% (24) of experts and affirm the model integrate such components and the 11% (3) indicates perhaps integrate the components. The results confirm the PCS include scientific, software and simulation components, which allow represent a SSD and simulating events (See Fig. 5).

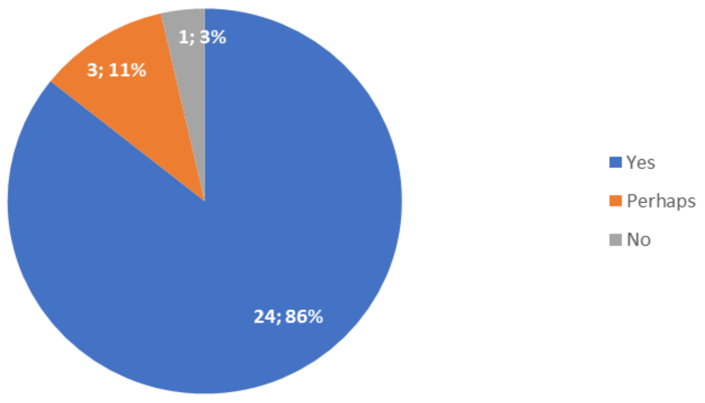

Fig. 5. Component integration

PCS understand. Experts answered the question: *Q3. Is the PCS understandable?* for analyzing the PCS understand variable. From the point of view of the experts, 20 experts are strongly agreed and agreed with the PCS is understandable, 7 experts are neutrals, and 1 expert is disagreed (See Fig. 7). The majority opinion of the results in show acceptation in the understanding of the schema.

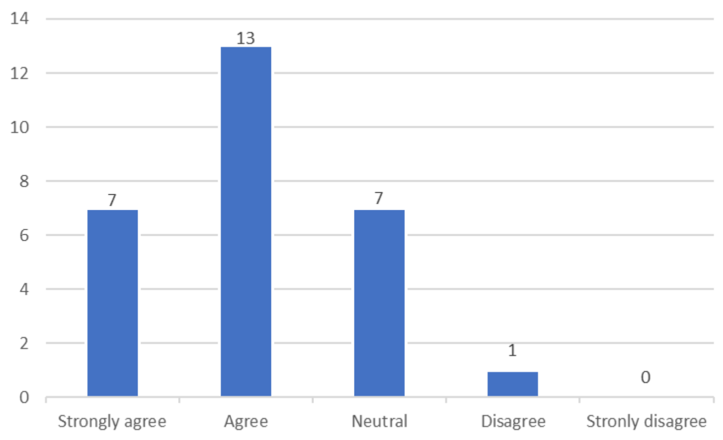

Fig. 6. PCS understand

PCS usage. Experts answered the question: *Q4. Would you use the PCS in scientific software development? e.g., in modeling, simulation and requirements elicitation* for analyzing the PCS usage variable. The 72% (20) experts affirm they could use the PCS in scientific software development processes, the 21% (6) experts indicate perhaps would use it and the 7% (2) answer they would not use it (See Fig. 7). The results present a positive position in the PCS usage for modeling SSD, event simulation, and requirements elicitation in scientific software development processes.

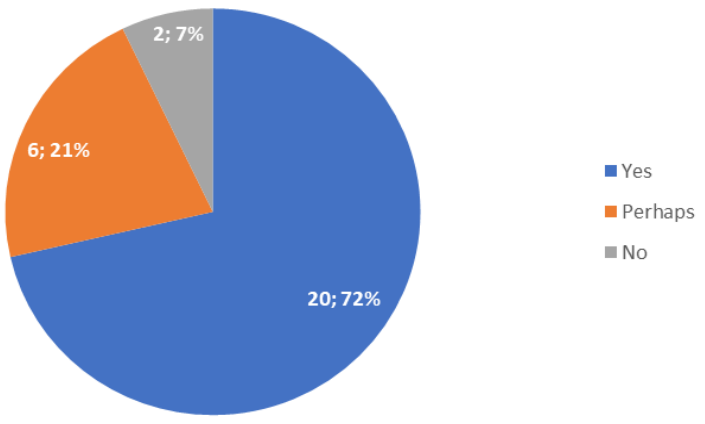

Fig. 7. PCS usage

6. CONCLUSION

Initial conditions, timers, class concepts, leaf concepts, events, dynamic relationships, variables, and *parameters* are identified in pre-conceptual schemas as components (elements) for simulating events. Such a representation was refined by defining *initialization, clock generation, random variable generation,* and e*vent generation* modules; such modules were proposed for completing components of event simulation in requirements engineering. PCS allow for modeling and simulating events in SSD by integrating simulation and software components in the PCS notation. Such components are applied in a lab study of an early earthquake alert generation system in the SSD of Geology. Event simulation by using the PCS is a new approach for integrating scientific and software components in the same model. Thus, scientists and software analysts can use the PCS as a computing model for modeling and simulating events in RE for the scientific software development process: they can validate the requirements with scientific stakeholders by using PCS, understand the elements of the complete system, and simulate events and data previously to the development phase of the scientific software. We also evaluate the understanding and the usage the PCS components by the expert opinion was favorable and both scientists and software experts understanded the model. This approach offers a better understanding of real-world events in any domain and consistency among the modeling, simulation, and implementation phases reducing the gap between science and software engineering.

We suggest as future work the automation of pre-conceptual schemas for simulating events, which allows for automatically generating results and reports from a simulation software.

ACKNOWLEDGEMENTS

This paper is a 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 Minciencias (Ministry of Science, Technology, and Innovation) from Colombia in the program Ph.D. students in Colombia, bid 727 of 2015.

REFERENCES

1. Bonaventura, M. A., and Castro, R. D. (2015). Ingeniería guiada por Modelado y Simulación de Eventos Discretos: Metodología y Caso de Estudio en la Red de Datos del Experimento *Tech. Rep.,* ATL-DAQ-PROC-2015-024.
2. Brosig, F., Meier, P., Becker, S., Koziolek, A., Koziolek, H., and Kounev, S. (2014). Quantitative evaluation of model-driven performance analysis and simulation of component-based architectures. *IEEE Transactions on Software Engineering*, 41(2), 157–175.
3. Camejo, I. M., Sailema, G. L. A., Carrillo, K. M. G., and Verdecia, J. A. M. (2018). Computational Simulation Model of Milk Production Process, Case Study: Dairy Plant FCP-ESPOCH. *KnE Engineering*, 179–191.
4. Colombian Geological Services Journals, https://www2.sgc.gov.co/Paginas/servicio-geologico-colombiano.aspx.
5. Durango, C.E., Noreña, P.A., Zapata, C.M. (2018). Representación de eventos de ruido ambiental a partir de esquemas preconceptuales y buenas prácticas de educción geoespacial de requisitos. *Research in Computing Science*, 147(2), 327–341.
6. García, A., Ortega, M., Izquierdo, D. (2012). *Elementos de simulación un enfoque práctico con witness*. Universidad Politécnica de Madrid: Madrid.
7. González, A., Luna, C., Abella, R. (2016). UML state machine as modeling language for DEVS formalism. *XLII Latin American Computing Conference* (CLEI), IEEE, 1–12.
8. González, A., Luna, C., Cuello, R., Perez, M., Daniele, M. (20140). Metamodel-based transformation from UML state machines to DEVS models. *XL Latin American Computing Conference* (CLEI), IEEE, 1–12.
9. Johanson, A. & Hasselbring, W. (2018). Software Engineering for Computational Science: Past, Present, Future. *Computing in Science & Engineering*, 20(2), 90–109.
10. Kanewala, U.E., Bieman, J. (2014). Testing scientific software: A systematic literature review. *Information and software technology*, 56(10), 1219–232.
11. Kelly, D. (2015). Scientific software development viewed as knowledge acquisition: Towards understanding the development of risk-averse scientific software. *Journal of Systems and Software*, 109, 50–61.
12. Laurindo, M. Peixoto, T., de Assis, J. (2019). Communication mechanism of the discrete event simulation and the mechanical project software for manufacturing systems. *Journal of Computational Design and Engineering*, 2019, 6(1), 70–80.
13. Li, Y., Guzman, E., Tsiamoura, K., Schneider, F. Bruegge, B. (2015). Automated requirements extraction for scientific software. *Procedia Computer Science*, 51, 582–591.
14. Luckham, D. (2011). *Event processing for Business: Organizing the Real-time Enterprise*. New Jersey: John Wiley & Sons.
15. Luckham, D. (2002). *The Power of Events. An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Boston: Addison-Wesley, 2002.
16. Noreña, P.A., Zapata, C.M. (2018). Una representación basada en esquemas preconceptuales de eventos determinísticos y aleatorios tipo señal desde dominios de software científico. *Research in Computing Science*, 147(2), 207–220.
17. Noreña, P.A., Zapata, C.M. (2018). A pre-conceptual-schema-based representation of time events coming from scientific software domain. *22nd World Multi-Conference on Systemics, Cybernetics and Informatics* (WMSCI), 53–58.

18. Noreña, P.A., Zapata, C.M., Villamizar, A.E. (2019). Representing chemical events by using mathematical notation from pre-conceptual schemas. *IEEE Latin America Transactions*, 17(01), 46–53.
19. Noreña, P.A., Zapata, C.M. (2019). Business Simulation by Using Events from Pre-Conceptual Schemas. *Developments in Business Simulation and Experiential Learning* 2018, 46, 258–263.
20. Noreña, P.A. (2020). An Extension to Pre-conceptual Schemas for Refining Event Representation and Mathematical Notation. Ph.D. Thesis, Universidad Nacional de Colombia, Medellín Campus, Colombia.
21. Rodríguez, J.M. Serrano, D., Monleón, T., Caro, J. (2008). Los modelos de simulación de eventos discretos en la evaluación económica de tecnologías y productos sanitarios.' Gaceta Sanitaria 2008, 22, 151–161.
22. Rus, I., Neu, H., Münch, J. (2014). A systematic methodology for developing discrete event simulation models of software development processes. arXiv:14033559 [csSE], https://arxiv.org/abs/1403.3559.4
23. Song, C., Guo, L., Wang, N. Ma, L. (2014). A grey box testing method for availability simulation software based on event tree model. *Prognostics and System Health Management Conference (PHM-2014 Hunan),* IEEE, 368–372.
24. Velásquez, S.: 'Un modelo ejecutable para la simulación multi-física de procesos de recobro mejorado en yacimientos de petróleo basado en esquemas preconceptuales.' M.Sc. Thesis, Universidad Nacional de Colombia, Medellín Campus, Colombia, 2019.
25. Zapata, C.M. (2012). *The UNC-Method Revisited: Elements of the New Approach*. Saarbrücken: Lambert Academic Publishing, 2012.
26. Zapata-Jaramillo, C. M., Zapata-Tamayo, J. S., & Noreña, P. A. (2020). Conversión de eventos desde esquemas preconceptuales en código PL/pgSQL: simulación de software en la cuarta revolución industrial. *Revista Ibérica de Sistemas e Tecnologias de Informação*, (39), 18–34.