UNIVERSIDAD NACIONAL DE COLOMBIA

An Extension to Pre-conceptual Schemas for Refining Event Representation and Mathematical Notation

Paola Andrea Noreña Cardona

Universidad Nacional de Colombia
Facultad de Minas, Departamento de Ciencias de la Computación y la Decisión
Medellín, Colombia
2020

An Extension to Pre-conceptual Schemas for Refining Event Representation and Mathematical Notation

Una extensión a los esquemas preconceptuales para el refinamiento en la representación de eventos y la notación matemática

Paola Andrea Noreña Cardona

Thesis presented as requirement for obtaining the title of:
Doctor en Ingeniería-Sistemas e Informática

Advisor:
Carlos Mario Zapata Jaramillo, Ph.D.

Jurors:
Luz Marcela Ruiz Carmona, Ph.D. Zurich University of Applied Sciences, Switzerland.
Óscar Pastor, Ph.D. Universidad Politécnica de Valencia, Spain.
Fernán Villa Garzón, Ph.D. Universidad Nacional de Colombia-Medellín Campus, Colombia.

Research Line:
Software Engineering
Research Group:
Computing Languages

Universidad Nacional de Colombia
Facultad de Minas, Departamento de Ciencias de la Computación y la Decisión
Medellín, Colombia
2020

Dedication

This Ph.D. Thesis is dedicated with immense love: to God for being my strength every day and giving me wisdom, knowledge, creativity, inspiration, and understanding; to my husband Luis Fernando for his patient, love, time, and courage; to my Mom for her support, help, advice, and confidence; and to my family for their hope to this work.

Acknowledgments

We must find time for stopping and thanking the people who make a difference in our lives.
—John F. Kennedy

Some people and institutions have contributed for constructing this Ph.D.Thesis.

I am very grateful to Minciencias (Ministry of Science, Technology, and Innovation) at Colombia in the National Ph.D. Student program, bid 727 of 2015 for sponsoring this Ph.D. Thesis during four years. Thanks to the Computing Languages research group, Departamento de Ciencias de la Computación y la Decisión, and Vicedecanatura de Investigación y Extensión at Facultad de Minas, Universidad Nacional de Colombia; the GIISTA (software engineering research group) at Tecnológico de Antioquia, Institución Universitaria; and the Middleware research group at the University of Toronto via the International Visiting Graduate Student program for supporting the time and resources to develop the research projects in the framework of this Thesis.

I am greatly thankful to my Ph.D. Advisor Carlos Mario Zapata Jaramillo who I admire profoundly for his knowledge, love, and passion to his work and family and who I consider an example to follow. I really appreciate and value his dedication, effort, patience, confidence, teaching, and advice.

I also express my sincere thanks the following people:

My jurors Luz Marcela Ruiz Carmona at Zurich University of Applied Sciences, Óscar Pastor at Universidad Politécnica de Valencia, Fernán Alonso Villa at Universidad Nacional de Colombia, and Liliana González Palacio at Universidad de Medellín for all their support, time, and orientation in my Ph.D. process. I especially thank Professor Marcela for accompanying me from the beginning of such a process.

My internship supervisor Arno-Hans Jacobsen at the University of Toronto for giving me the opportunity to work in the Middleware research group, teaching me his simplicity, and guiding me in my immersion and new knowledge of the complex event processing. Middleware research group members: Bogdan Scaunasu, Geoffrey Elliott, Fei Pan, and Edward Zhang for their helpful, knowledge, explanation, and correction in my Internship Project and my English progress; and professor Patricia Vilain for her suggestions in my internship presentation and validation. Erika Londoño, Maria Eugenia Restrepo, and their family for their hospitality, help, and company in Canada.

Professors: Luca Cernuzzi at Universidad Católica de Asunción for their recommendations and questions to the Ph.D. Thesis proposal; Bell Manrique at Universidad de Medellín for being an example to my doctorate and guiding me in the experimental validation in software engineering; Gloria Gasca at Universidad de Medellín for her suggestions in contributions and findings; and Albeiro Espinosa at Universidad Nacional de Colombia for his help and advice.

My university partners: Johnathan Calle for his effort for working with me in the mathematical notation refinement of the pre-conceptual schemas, reading, and reviewing my advances of this document; Steven Velásquez for representing the oil well domain and complex mathematical equations in his M.Sc. Thesis by using our proposed extension and helping me in the form aspects of this document; Sebastián Zapata for his suggestions in the chemical domain representation and for using such a representation in his M.Sc. Thesis; and Rubén Sánchez for his explanations and suggestions in the electrical domain representation. Diana Medina, José David Mosquera, and other undergraduate students in the course requirements engineering (2017-2) at Universidad Nacional de Colombia and Carolina Cárdenas and Daniel Bedoya undergraduate students (2018) at Tecnólogico de Antioquia for representing several events in scientific software domains, and Aixa Villamizar for allowing me to work in the research project for the Tecnólogico de Antioquia. Samir Guarín for his suggestions in medical and chemical domains.

All participants of the experimental validation for their suggestions, time, attention, and interest. I especially thank ABSEL (The Association for Business Simulation and Experiential learning) members. José David Mosquera for participating in the validation by using the software development and simulation of a pre-conceptual schema represented in an industrial domain.

My colleagues and friends: Diana Maria Torres Ricaurte for teaching me about technical events in interoperability of systems; Claudia Elena Durango Vanegas for teaching me about natural events in geographic information systems; Grissa Vianney Maturana for helping me in the event games and class processes; Maria Clara Gómez for playing my event game and giving me suggestions; Juan Ricardo Cogollo for explaining me technical concepts; Jesus Insuasti for philosophizing with me, and Wilder Perdomo for his interest. I highly value their company, support, teaching, explanation, and advice for me in our Ph.D. process. Fernanda Guidotti (at Universidade de São Paulo) for teaching me Latex; Ivonne Rodríguez, Jenny Ríos, Laura Ardila, and Jose Daniel Morcillo for their suggestions and interest. I appreciate the friendship we have constructed.

All the anonymous reviewers of our papers and chapters and the jurors of our presentations for their comments and feedback.

Abstract

An event is an occurrence within a particular software system or domain. Software and scientific models are representations of computing and natural systems. Such models have software and scientific components—domain knowledge elements. Scientists and business analysts use such models and their components for recognizing a domain, *e.g.,* pre-conceptual schemas (PCS) used in software engineering. Scientific software domains (SSD) comprise fields in engineering and science, which are focused on developing and simulating scientific software systems for event or phenomenon research. Event-based software development has increased in scientific domains. Approaches to event-driven modeling are used from software/scientific modeling. Some advances have emerged in such approaches for integrating software and scientific components in science and engineering projects. However, scientists and business analysts lack a computational model for SSD in order to integrate both components in the same model. PCS notation includes software components based on structural and dynamic features, which allow for representing events and mathematical operations. Nonetheless, PCS lack scientific components for representing events in SSD. In this Ph.D. Thesis, we propose an extension to pre-conceptual schemas for refining event representation and mathematical notation. Such an extension comprises scientific components as graphical, linguistic, and mathematical structures for the sake of such refinement. We validate our proposal by using both an experimental process and a software application. Extension to PCS is included as a new work product for representing events in SSD. Therefore, the extended PCS are intended to be computing models for scientists and business analysts in scientific software development and simulation processes.

Keywords: Computational Science and Engineering projects, Event Representation, Mathematical notation, Pre-conceptual Schemas, Software Engineering, Software Modeling, Scientific Software Systems, Scientific Software Domains, Simulation.

Resumen

Un evento es una ocurrencia en un sistema de software o dominio particular. Los modelos científicos y de software son representaciones de sistemas informáticos o naturales. Esos modelos tienen componentes científicos y de software (elementos del conocimiento del dominio). Científicos y analistas de negocio usan estos modelos y sus componentes para reconocer un dominio. Un ejemplo de esos modelos son los esquemas preconceptuales (EP), que se usan en ingeniería de software. Los dominios de software científico comprenden áreas en ingeniería y ciencia que se enfocan en el desarrollo y simulación de sistemas de software científico para la investigación de eventos o fenómenos. El desarrollo de software dirigido por eventos se viene incrementando en dominios científicos. Enfoques de modelado basado en eventos se usan desde el modelado científico y el modelado de software. En estos enfoques surgen algunos avances para integrar componentes científicos y componentes de software en proyectos de ingeniería y ciencia. Sin embargo, científicos y analistas de negocio carecen de un modelo computacional para dominios de software científico que integre ambos componentes en el mismo modelo. La notación de los EP incluye componentes de software que se basan en características estructurales y dinámicas, los cuales permiten representar eventos y operaciones matemáticas. No obstante, los EP carecen de componentes científicos para representar eventos en dominios de software científico. En esta Tesis Doctoral se propone una extensión a los esquemas preconceptuales para el refinamiento en la representación de eventos y la notación matemática. Esta extensión integra componentes científicos (estructuras gráficas, lingüísticas y matemáticas) para lograr este refinamiento. También, se valida la propuesta mediante un proceso experimental y una aplicación de software. La extensión a los EP se incluye como un nuevo producto de trabajo para representar eventos en dominios de software científico. Por lo tanto, se pretende que los EP extendidos sean modelos de computación, para científicos y analistas de negocio en procesos de desarrollo y simulación de software científico.

Keywords: científicos, dominios de software científico, esquemas preconceptuales, ingeniería de software, modelado de software, notación matemática, proyectos de ingeniería y ciencias computacionales, representación de eventos, sistemas de software científico, simulación.

Content

List of Figures

List of Tables

1 Introduction

I am more and more convinced that our happiness or unhappiness depends far more on the way we meet the events of life, than on the nature of those events themselves.

—Wilhelm von Humnboldt

Events are something that happens in either the real world and in a software system or domain, which are also called phenomena (Ravikumar *et al.*, 2016; Liu *et al.*, 2016). Events are used for driving business and automated processes (Luckham, 2011). The word *event* is also used for naming a programming entity or object as an occurrence in a computing or natural system (Etzion *et al.*, 2011). The event functionality implies its internal logic conformed by conditions and operations (Vásquez & Sandova, 2017; Wonham *et al.*, 2018). A *trigger* is an event used for beginning processes (OMG, 2011, 2015), *e.g*, volcano erupts, sensor starts, patient suffers heart attack, and earthquake arrives (Noreña *et al.*, 2018; Noreña & Zapata, 2018a).

Models are used in science and engineering for providing abstractions of a system (Gomaa, 2011). Software models are used as a central tool in the software engineering process in computing systems (Gomaa, 2011; Da Silva, 2015). Scientific models are used for understanding events in natural systems (Gilbert, 2004). Processes, events, concepts, and structures—graphical, linguistic, and mathematical—are scientific and software components, domain knowledge elements of a model (Gilbert, 2004; Haas, 1960; Jaramillo & Esteban, 2006). Scientists and business analysts use such models and their components in order to obtain a better understanding of the knowledge domain (Boubeta-Puig *et al.*, 2015, 2019).

Pre-conceptual schemas (PCS) are models used in software engineering for linguistically and graphically recognizing a domain. PCS notation is defined by applying the computational linguistics rules, so analysts and stakeholders easily understand the domain knowledge and developers can consistently code the software system (Zapata, 2012; Noreña & Zapata, 2018b). Scientific software domains (SSD) include fields in engineering and science for studying an event or phenomenon. Such studies are obtained by developing software systems with scientific knowledge based on mathematical models and simulating results from science and engineering projects (Kelly, 2015; Li, 2015), *i.e.*, chemistry, physics, biology, mathematics, statistics, environmental sciences, electronics, petroleum engineering, medicine, geography, meteorology, geology, etc. (Wiese *et al.*, 2019).

Applicability of events in the development of software systems is increasing (Campos-Rebelo *et al.*, 2015; Luckham, 2011). Computer systems are driven by events and this is a reason for such applicability (Luckham, 2002). Discrete event simulation languages (domain-specific languages) and simulators (Johanson & Hasselbring, 2018; Li, 2015), network development, active databases, middleware, event-driven architecture (EDA), and strategic management (event-driven modeling, business intelligence, and complex event processing) are event-driven trends (Luckham, 2011).

Event applicability motivates our review about event-driven modeling from software/scientific modeling. Business process model notation (BPMN), unified modeling language (UML) activity, state machine, and sequence diagrams, Medit4CEP tool, etc. are software modeling approaches used for analyzing the structural or behavioral view of a system (OMG, 2011, 2014a, 2015; Chonoles, 2017; Boubeta-Puig *et al.*, 2015, 2019; Haisjackl *et al.*, 2018). Petri net, finite automaton, Markov model, block diagram, etc. are scientific modeling approaches used for analyzing mathematical models of an event (Chen *et al.*, 2017; Liu & Zhao, 2016; Luo & Zhou, 2016; Sarno *et al.*, 2015; Wang *et al.*, 2017, 2018; Zhang & Zhang, 2016; Zhong & He, 2016).

BPMN, UML diagrams, ontologies, frameworks, Petri nets, etc. are used for integrating software components—concepts (classes, attributes), processes, events, and structures—and scientific components—mathematical structures, scientific concepts or terminology, processes, and events—in science and engineering projects (Bazhenova *et al.*, 2019; Bazydlo *et al.*, 2014; Gao *et al.*, 2014; Li, 2015; Li *et al.*, 2015; Patri *et al.*, 2014; Sahoo *et al.*, 2015; Xia *et al.*, 2019). Some event representations (Zapata, 2012; Noreña, 2014; Zapata *et al.*, 2013, 2014) and mathematical notation for organizational domains (Chaverra, 2011) and scientific software (Calle, 2016) have been proposed in PCS. Some of such approaches also present advances of linguistic structures for representing events (OMG, 2011, 2014a, 2015; Boubeta-Puig *et al.*, 2015, 2019; Zapata, 2012).

Such approaches are attempts for addressing the gap between the event-driven software modeling of software engineering and event-driven scientific modeling of science (Johanson & Hasselbring, 2018). However, problems still arise since scientific modeling lacks software components and software modeling lacks scientific components. Thus, both scientists and business analysts require a computing model with integrated scientific and software components for event representation and mathematical notation in SSD (Johanson & Hasselbring, 2018; Kanewala & Bieman, 2014; Wiese *et al.*, 2019; Wilson *et al.*, 2014).

Advances on the PCS notation allow for including software components (processes, events, concepts, and structures) for representing events and mathematical notation in the same model (Zapata, 2012; Noreña, 2014; Zapata *et al.*, 2013, 2014; Chaverra, 2011; Calle, 2016).

However, PCS lack scientific components for representing events in SSD, since graphical, linguistic, and mathematical structures are insufficient for representing events (time events and others) and their functionality—which is established by integrating scientific components from SSD. Such a representation is required for the analysis, development, and simulation of scientific software.

Consequently, in this Ph.D. Thesis we integrate software (processes, events, concepts, and structures) and scientific components (graphical, linguistic, and mathematical structures) in the same model, the so-called *pre-conceptual schemas*. Thus, new structures are defined as extensions to PCS in order to refine the event representation and mathematical notation in SSD. Such an extension is achieved by characterizing events emerging from SSD and defining new mathematical, linguistic, and graphical structures for representing events in PCS.

Finally, we perform an experimental validation with scientific, software, and simulation experts for evaluating the understanding about the extended PCS and the mathematical notation/event notation, and the usability of the PCS. Such validation is performed by using the experimental process of software engineering: planning, executing, and analyzing experiment (Wieringa, 2014; Wohlin *et al.*, 2012) and a software application for translating PCS to code.

This extension to PCS is a work product allowing scientists and business analysts for representing events/phenomena in SSD by using mathematical notation. The application of the extended PCS as computing models is intended to produce the following contributions: (i) structural and dynamic representation of the elements of any SSD; (ii) time and functionality representation of the events in SSD; (iii) understanding and recognizing of the processes, events, concepts, and mathematical models present in a SSD; and (iv) PCS usability for developing and simulating scientific software systems.

This Ph.D. Thesis is structured as follows: in Chapter 2 (background) we present the conceptual framework, Ph.D. Thesis focus, and methodology; in Chapter 3 (research problem) we indicate our motivation, state of the art, problem statement, research question, hypothesis, objectives, and justification; in Chapter 4 (extension to PCS) we propose a work product for representing events and mathematical notation in SSD by using graphical, linguistic, and mathematical structures in PCS; in Chapter 5 (validation) we experimentally evaluate the understanding and usability of our proposal, apply the PCS to a programming language, and present the publications related to this Ph.D. Thesis, other representations, and PCS templates to be used in case tools; in Chapter 6 (conclusions and challenges) we define the contributions and future work.

2 Background

Knowledge is having a mental history of past events and wisdom is having the ability for relating those past events to the present and future.

—Steven Magee

2.1. Conceptual Framework

2.1.1. Events

According to the common-sense meaning in the dictionary, an *event* is something that happens (Luckham, 2011). Such an event occurs within a particular system or domain (Etzion *et al.*, 2011; Ravikumar *et al.*, 2016; Liu *et al.*, 2016). An event is also a programming entity or object, which represents such an occurrence in the system (Etzion *et al.*, 2011). Information systems are driven by events (Luckham, 2002). Thus, events are used for driving business and automated processes (Luckham, 2011) and controlling the system behavior, since they are responsible for changing the process state. Such changes are produced by using constraints from the events (Noreña & Zapata, 2018b,c).

Event functionality is the specification of the internal logic of events. Such a logic contains conditions and mathematical operations for controlling and understanding the behavior of the system. A change of states in the system occurs when the logic of an event is executed in a sequence of time. An analyst should define such a logic for modeling a system (Vásquez & Sandova, 2017; Wonham *et al.*, 2018).

Trigger concept is used for indicating an event generating the start of processes, process flows, services, and other events, which is commonly used in databases (OMG, 2014a, 2015). Trigger events are classified as:

A *none* or *statement* is an instruction used for activating a trigger, which generates a process. *e.g.*, when a signal starts, a message or an error emerges.

A *conditional* is a constraint used for specifying a condition. Such an event is triggered when an state changes, *e.g.*, temperature above 300°C is true.

A *timer* is a specific time or a cycle used for triggering the start of a process, *e.g.*, every Monday at 9 am (OMG, 2014a).

Some examples of trigger events are: if someone is working on a laptop in a coffee shop, and a robbery happens, such a robbery event would disrupt the peaceful atmosphere and compel people to react (Etzion *et al.*, 2011); a phone rings, an email arrives, an alarm sounds, and a sensor signal starts are events occurring from devices; a volcano erupts, an earthquake occurs, a hurricane appears, and noise environmental increases are natural events; a patient suffers a heart attack and an animal bleeds are chemical events occurring in living beings (Noreña *et al.*, 2018; Noreña & Zapata, 2018a; Noreña *et al.*, 2019; Durango *et al.*, 2018).

2.1.2. Model

Modeling is a well-known technique adopted by science and engineering fields for providing abstractions of a system/domain (Da Silva, 2015; Gomaa, 2011). A *model* is constructed for analyzing and understanding such abstractions (Boubeta-Puig *et al.*, 2015, 2019). A model allows for sharing a common vision and knowledge of such a system (Da Silva, 2015), *e.g.*, physical, mathematical, biological, and technical models.

Modeling is used for designing computing systems before coding them in the software development process. A *software model* (also called conceptual model/process model) plays a role in such a process for fulfillment business functionality, end-user achievement need, program design, and requirements, before coding the system (OMG, 2011; Gomaa, 2011; Da Silva, 2015). Such models contain software components for recognizing a domain. Model-driven engineering (MDE)—a software engineering paradigm—includes the use of models as documentation, work products, and tools in engineering disciplines and any application domain. Model-driven architecture (MDA) is an approach from MDE for deriving value from models (OMG, 2014b). Perspective dimension of a model is classified by MDA as structural, behavioral, and multiple (Da Silva, 2015; Giraldo *et al.*, 2019). A *structural* (static) view is used for describing a system from its structural perspective by using concepts such as classes, objects, nodes, blocks, and respective relationships, *e.g.*, ontologies as a comprehensive event ontology (CEVO; Shekarpour *et al.*, 2019), unified modeling language (UML) class and component diagrams. A *behavioral* (dynamic) view is used for describing the behavior of a system by using operations, processes, states, and events, *e.g.*, event-driven process chain (EPC; Xue *et al.*, 2013). A *multiple* view is used for including both static and dynamic views (Da Silva, 2015).

Modeling in science is used for producing, disseminating, and accepting scientific knowledge from natural systems. Such knowledge is described by using scientific models, which are based on mathematical models (which are discretized, *i.e.*, process for transferring continuous to discrete functions) and scientific concepts/terminology. A *scientific model* is used for analyzing entities or objects and their relationships, *e.g.*, entities of the organs of the human body, of an electric motor; complex phenomena/events in a time segment of behavior of a system, *e.g.*, a block diagram (Zhong & He, 2016). Design and experimental practices are based on computing models (Gilbert, 2004). A *computing model* is used for analyzing the behavior of a complex system by using computer simulation, *i.e.*, an algorithm, mathematical or graphical model. A computing model contains numerous variables and an set of instructions for characterizing the system (Kanewala & Bieman, 2014).

Components

Processes (actions of the system), *events* (automated processes/phenomena), *concepts* (terminology), and *structures* (symbols) are common software and scientific model components. The following structure types are identified in such models:

Graphical structures are organized symbol sets used for allowing a visual representation of the elements of a model, *e.g*, a hexagon symbol in EPC diagram is used for representing events (Xue *et al.*, 2013). Usually, the graphical structures should include linguistic/mathematical structures.

Linguistic structures—also linguistic units—are used for referring to terms of a sentence and their relationships (Haas, 1960), *e.g*, "new repair task arrived" is an event in an EPC diagram (Xue *et al.*, 2013). A *verb* is a linguistic structure used as the main term of a sentence, which may be compared to a sort of atom, susceptible to attracting a greater/lesser number of actants/arguments associated with the verb (Tesnière, 1965), *i.e.*, "arrived." A *semantic role* is a linguistic structure used for conceptually relating the verb and its arguments (Moreda, 2008; Payne, 1997). Semantic roles allow for analyzing syntactic and semantic relationships and identifying the function of the verb argument in an event, which is expressed by using such a verb (Moreda, 2008). Semantic roles are also referred to deep cases, thematic roles, and theta roles (Payne, 1997). Actants and circumstants are types of semantic roles, the immediate subordinates of the verb (Tesnière, 1965).

Actants are arguments necessarily used for completing the meaning of a given full verb. They are the beings/things (nouns) which participate in the process, performing/receiving the action. According to Tesnière (1965) a given verb is: *avalent*, a verb for denoting a meteorological phenomena without actants, *e.g.*, it rained; *monovalent*, a verb for expressing an action which only a single person or thing participates, *e.g.*, Amy slept; *divalent*, a verb with two actants for expressing an action which two people or things participate, *e.g.*, Amy met Paola; and *trivalent*, a verb with three actants for expressing an action which three people or things take part, *e.g.*, Amy gave Paola a chocolate. Fillmore (1977) and Gruber (1965) define actant type-semantic roles as *agent*, who performs the action, *e.g.*, engineers select nearest engineer (Xue *et al.*, 2013); *experiencer*, which experiments an action or event, *e.g.*, new repair task arrived (Xue *et al.*, 2013), an airplane falls; *patient*, who suffers the event effect, *e.g.*, the baby (first actant, patient) suffers dizziness (second actant); and *beneficiary*, who receive the benefit of an action, *e.g.*, the professor teaches the student; in this sentence, the professor is an agent and the student is a beneficiary (Noreña *et al.*, 2018).

Circumstants are adjuncts, circumstancial complements, and adverbial functions used for extending the meaning of the verb. Circumstants can express the circumstances in which a process takes place (Tesnière, 1965). Gruber (1965) define circumstant type-semantic roles as *strength*, whose origin is unknown and produced by an event, *e.g.*, rock melts; and *cause*, situation generated by an event, *e.g.*, the baby suffers dizziness. We can perceive the articulation of a real-life experience with linguistic structures by understanding a sentence with a verb as a node and its connections with both actants and circumstants, so the event is structured by using the language. Such an experience, *i.e.*, a process/event, actors, and circumstances can be transferred to structural syntax, and then applied by using a verb, its actants, and circumstants (Tesnière, 1965).

Mathematical structures are sets of related mathematical objects as concepts, operators, relationships, and rules expressed in equations for solving operations (Jaramillo & Esteban, 2006). Mathematical operators can be: *logical* when they are used in conditions for joining relational operators, *i.e.*, and, or, xor; *relational* when they are used in conditions, *i.e.*, less than (<), greater than (>) , equal (=), not equal (!=), etc; *basic* when they are used in arithmetic operators, *i.e.*, plus (+), minus (-), multiplication (*), division (/); and *complex operators* used in complex equations, *i.e.*, trigonometric functions as sine (*sin*) and cosine (*cos*), exponential function (*Exp, e*), logarithmic function (*log, ln*), etc.

2.1.3. Analyst

A *scientist* can make predictions with the simulation results about what could happen in the real system for finding the solution to the problem, *e.g.*, if a building lacks the right concrete mix, a disaster can occur. Scientists use scientific models for analyzing whether the system is properly working and what events are emerging from the system (Kanewala & Bieman, 2014). Commonly, this is a mathematical analysis for reviewing if the model is correct according to the studied phenomenon (Kanewala & Bieman, 2014). Scientists also require such models for developing computing systems, coming from scripts for small-scale data analysis to complex coupled multiphysics simulations executed on high-end hardware (Johanson & Hasselbring, 2018).

A *business analyst* is a role of the software team during analysis and design phases of a software engineering process. Such a role uses software engineering techniques (Johanson & Hasselbring, 2018) and models for recognizing a domain and obtaining the requirements of a computing system. Analysts identify real-world objects in the problem domain and design the corresponding objects in the system model (Gomaa, 2011).

2.1.4. Pre-conceptual Schemas (PCS)

A *schema* is a model used in computational learning theory for understanding a declarative and procedural knowledge of a domain (Pozo, 2006). *Pre-conceptual* is a term used in philosophy and pedagogy; pre-concepts are used for constructing a concept by using previous knowledge; pre-conceptual phase comprises intuitive interpretations (pre-concepts) about the world to be used for conceptualizing them (Zapata, 2007). A *Pre-conceptual schema* is a graphical and conceptual model used in software engineering (Zapata, 2012; Noreña & Zapata, 2018b). PCS integrate an intuitive and pegagogical nature (Zapata-Tamayo & Zapata-Jaramillo, 2018), allowing users for understanding the main software components—concepts, processes, events, and structures—belonging to a domain (Zapata, 2012; Noreña & Zapata, 2018b). PCS also involve dynamic and structural features for creating complete and consistent view of a model (Zapata, 2012).

PCS Notation

PCS notation contains the following software components (see Figure **2-1**) organized in four groups (Zapata, 2012):

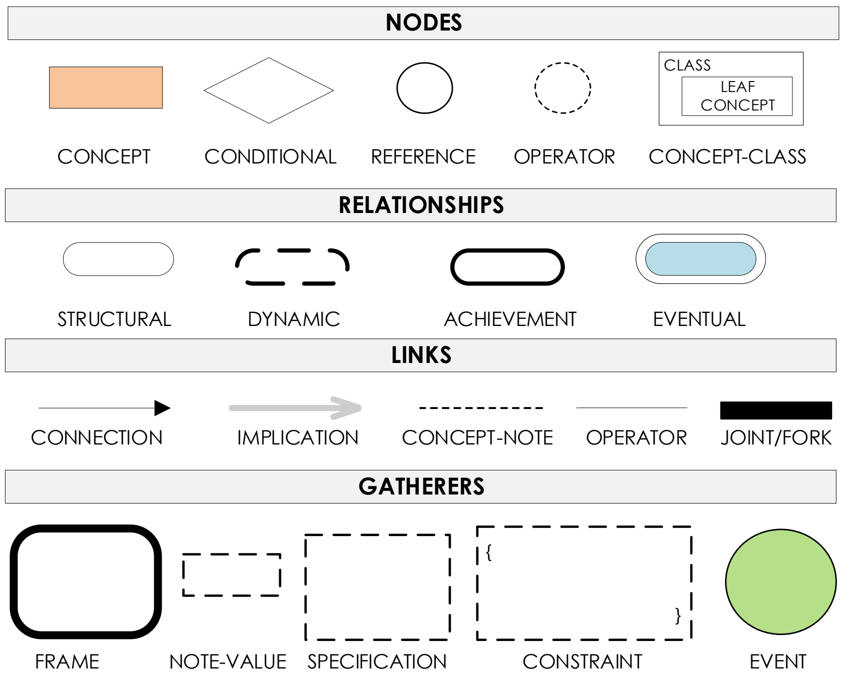


Figure 2-1 PCS Notation. The Authors based on Zapata (2012)

Nodes

A *concept* is used for representing a class concept and a leaf concept/attribute, *e.g.*, seismologist, sensor, medical history.

A *conditional* is used for defining an instruction, *e.g.*, if alarm = on.

A *reference* is used for relating a distant node by using a number.

An *operator* is used for representing mathematical operations. Operators can be: *logical* (AND, OR); *basic* $(+, -, *, /)$; and *relational* $(<, <=, >, >=, =)$.

A *concept-class* is used for representing a class with its leaf concept *e.g.*, biology code.

Relationships

A *structural relationship* is used for relating a class concept and its leaf concepts by using the verb “has,” *e.g.*, biology has code, and defining an inheritance by using the verb “is,” *e.g.*, user is scientist, user is business analyst.

A *dynamic relationship* is used for representing a process/activity/service, *e.g.*, doctor reviews medical history, sound engineer measures noise.

An *achievement relationship* is used for representing objectives, *e.g.*, improving security, looking door.

An *eventual relationship* is used for representing events with a concept/noun and an eventual verb, *e.g.*, file arrives (Noreña, 2014).

Links

A *connection* is used for relating nodes and relationships.

An *implication* is used for relating dynamic relationships, conditionals, and events.

A *concept-note* is used for relating values, specifications, and constraints.

An *operator* is used for relating operators, concepts, and values.

A *joint/fork* is used for relating implication links.

Gatherers

A *frame* is commonly associated with reports.

A *note-value* is an assignation value of nodes.

A *specification* is used for including values and operations without conditions.

A *constraint* is used for including values and operations with conditions.

An *event* is used for triggering dynamic relationships and other events (Zapata, 2012; Noreña *et al.*, 2018).

PCS notation is defined by applying rules based on computational linguistics (discipline focused on formalizing the computer language from the language the natural) allowing analysts and stakeholders for understanding the main elements of the domain knowledge and developers for consistently coding a software system (Zapata, 2012; Noreña & Zapata, 2018b). Some linguistic rules are:

(i) Subjects and objects are nouns, then a *concept* structure should be used for representing them, *e.g.*, *cow* and *milker* in Figure **2-2**.

(ii) Every complete sentence contains a subject and a predicate (an object with a verb and another object), then every *dynamic* and *structural relationship* should have a triad including a *concept*, a *relationship*, and another *concept*, *e.g., cow has name* (structural relationship) and *milker collects milk* (dynamic relationship, see Figure **2-2**).

(iii) *Concepts* should be used in a singular form, *e.g., milker* and *seller* in Figure **2-2**.

(iv) Semantic roles allow for classifying a verb (verb categories, *i.e.*, state, activity/process, event, achievement), which are used for defining every relationship, then an agent-semantic role should be used in the first concept for representing a *dynamic relationship*, *e.g., milker collects milk* (*milker* is the agent role) while an *eventual relationship* should lack an agent-semantic role, since is different to a process, but an eventual verb should have from zero to one actants, *e.g., customer arrives* in Figure **2-2** (Noreña, 2014).

Some graphical rules are:

(i) *Concepts* and *relationships* should be linked to the *connection* link, *e.g.*, *connection* link between the concept *cow* and the dynamic relationship *produces* (see Figure **2-2**).

(ii) A *process* flow, an *event* flow, an *event*/a *conditional* related to a *dynamic relationships* should be linked to the *implication* (gray color) link and achievement flow with the *implication* (black color) link, *e.g.*, *implication* link between the event *customer arrives* and the dynamic relationship *seller sells milk* (see Figure **2-2**).

(iii) An *operator node*, a *note-value* (possible values of a concept), a *specification*, a *constraint*, and an *achievement relationship* should linked to the *concept-note* link, *e.g.*, the *note-value* related to the concept *amount* (see Figure **2-2**).

(iv) Mathematical operations (concepts and note-values) should be linked to the *operator* link (Zapata, 2007), *e.g.*, the *operator* links used in the *conditional milk.amount >= 30 Liters* (see Figure **2-2**).

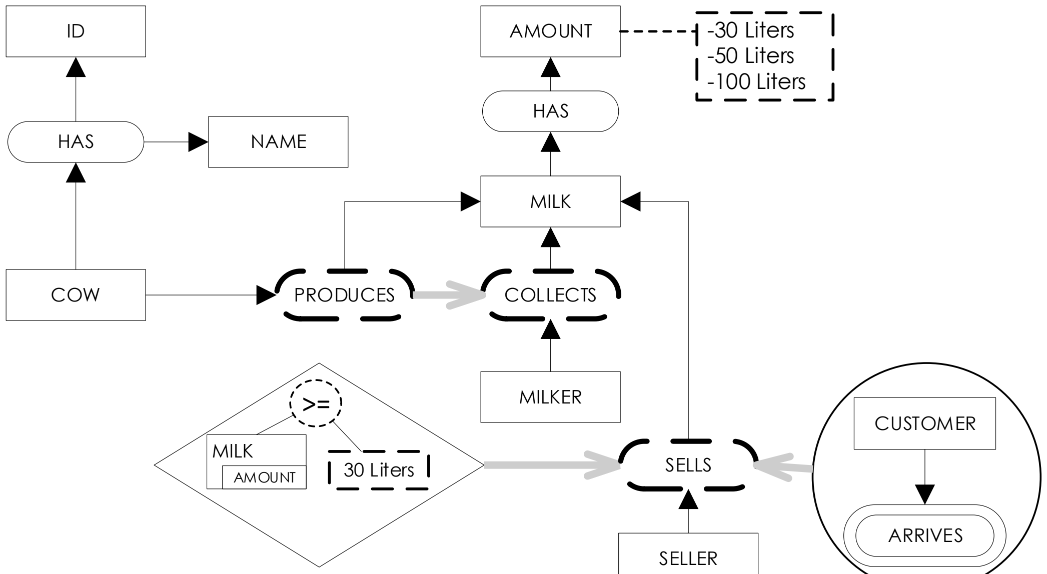


Figure 2-2 PCS Example. The Authors based on Zapata-Tamayo & Zapata-Jaramillo (2018)

2.1.5. Scientific Software Domains

Scientific software domains (SSD) include fields for developing scientific software systems (Kelly, 2015), *e.g.*, chemistry, physics, mathematics, statistics, economy, industry, environment, geography, science, biology, bacteriology, geology, vulcanology, meteorology, electronics, mechanics, medicine, and others (Wiese *et al.*, 2019). Scientific software systems are created by scientists and engineers (Heaton & Carver, 2015; Howison *et al.*, 2015; Johanson & Hasselbring, 2018; Wilson *et al.*, 2014) in science and engineering projects (Li *et al.*, 2015). Such systems are mainly developed for solving problems and research questions (Nanthaamornphong & Carver, 2017), improving the understanding of the behavior (Howison *et al.*, 2015), making predictions, increasing the knowledge about real-world processes and events/phenomena, and supporting critical decision making (Kanewala & Bieman, 2014; Kelly, 2015), *i.e.*, weather forecasting, global climate change, genomics, human health, etc. (Nanthaamornphong & Carver, 2017).

Some key features of the scientific software are (i) dynamic requirements (Nanthaamornphong & Carver, 2017), (ii) mathematical models, numerical methods, and physical phenomena, and (iii) domain experts, since scientists often develop scientific software themselves (Kanewala & Bieman, 2014) due to the complexity of the domain and the system (Calle, 2016; Kelly, 2015). Some examples of software systems are: software for studying the safe operation of nuclear plants, tracking paths of hurricanes, locating satellites in telescope images, checking mineshafts for rock faults, modeling medical procedures for cancer treatment, and studying ocean currents for ecological impact (Kelly, 2015).

Event types found in scientific software domains are: *natural events*, which happpen in natural systems, *e.g.*, enviromental noise increases (Durango *et al.*, 2018; Noreña *et al.*, 2018); *discrete events*, which happen in a specific time in dynamic system, *e.g.*, an alarm sounds every day 5am; *deterministic events*, which happen in predictable values, *e.g.*, signal emerges; and *non-deterministic events*, which happen randomly, *e.g.*, customer arrives (Noreña & Zapata, 2018a). Such event types are classified as trigger events.

2.2. Ph.D. Thesis Focus

We relate the conceptual framework and the Ph.D. Thesis focus in Figure **2-3** by using a PCS. This Ph.D. Thesis is focused on event-driven modeling by integrating two fields: science and software engineering. We specifically work on the software analysis (core process of software engineering) where the analyst, *i.e.*, a business analyst in software fields or a scientist in science fields uses a model for recognizing a domain (understanding its elements).

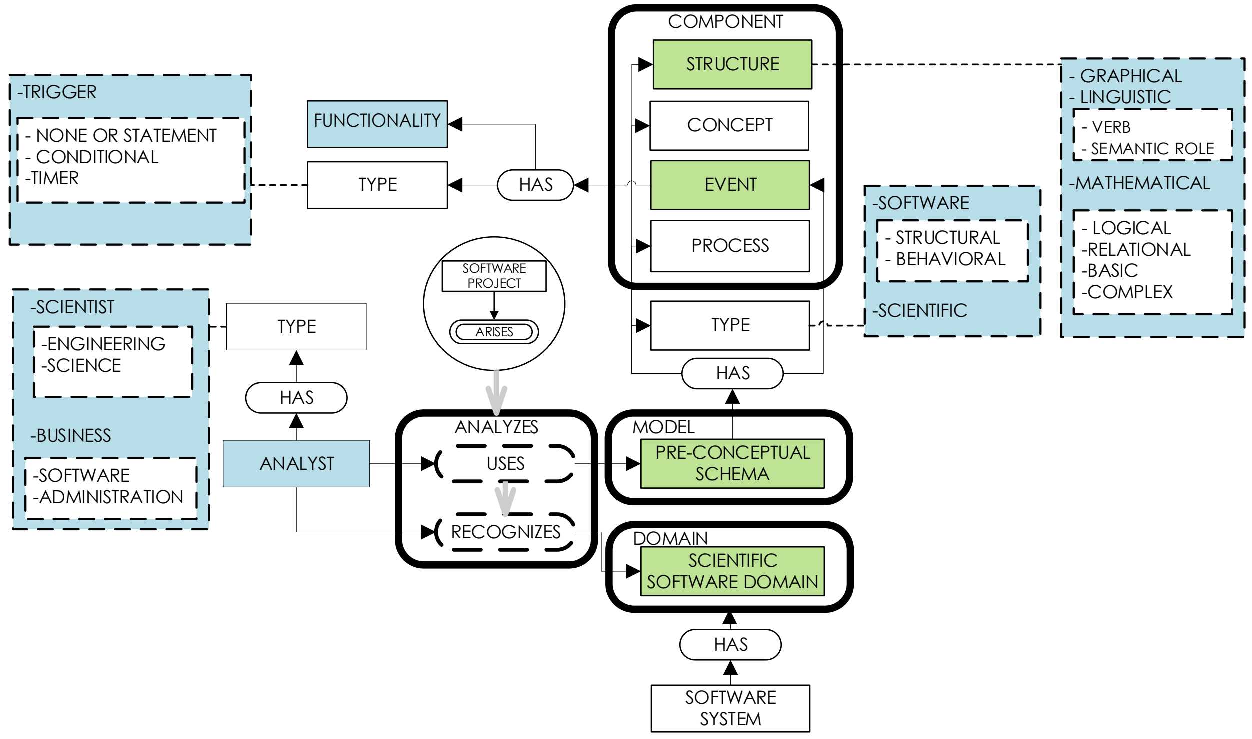


Figure 2-3 Conceptual framework. The Authors

In this Ph.D. Thesis we use pre-conceptual schemas as models. Perspective type is a multiple view (structural and behavioral), abstraction level is logical (models of the way the components of a system interact with each other and with people) according to the MDA (OMG, 2014b), and the domain used is the scientific software domain. Our research is aimed at the refinement of the event representation and mathematical notation in PCS. Thus, an extension to PCS is proposed by using graphical, linguistic, and mathematical structures for representing SSD (science and engineering fields) where trigger events are predominant.

2.3. Methodology

We define four phases by using the *empirically-based technology transfer methodology* (see Figure **2-4**) supported by the *experimentation in software engineering* (Wohlin *et al.*, 2012). Such a methodology is applied for sharing knowledge, new tools, technology, and methods between Academia and Industry.

In Academia, the problems are observed from the Industry. Solutions are proposed for both parts. Finally, we apply the *experimentation in the software engineering process* (Wohlin *et al.*, 2012) for validating the solution. This process is also considered in the *design science methodology for information systems and software engineering* (Wieringa, 2014).

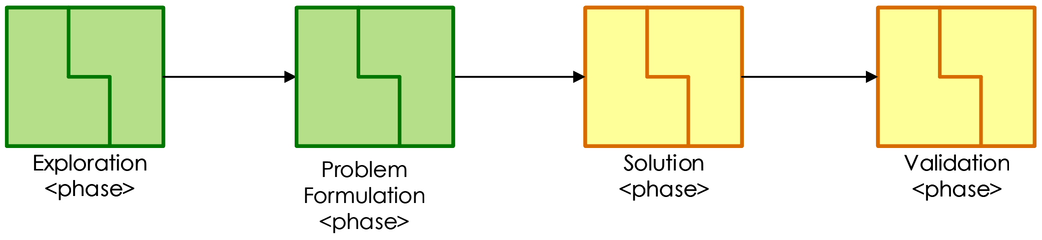


Figure 2-4 Research Methodology. The Authors

2.3.1. Exploration

A systematic literature review is carried out for synthesizing and analyzing the available evidence related to the research in a scientific and rigorous way (Wohlin *et al.*, 2012). Such a review is based on the guidelines for software engineering proposed by Kitchenham and Charters and supported by the experimentation process in software engineering (Wohlin *et al.*, 2012). *Planning literature review* and *executing systematic literature review* activities are developed for obtaining the *review protocol, background, primary studies, list of studies*, and *study analysis* (see Figure **2-5**).

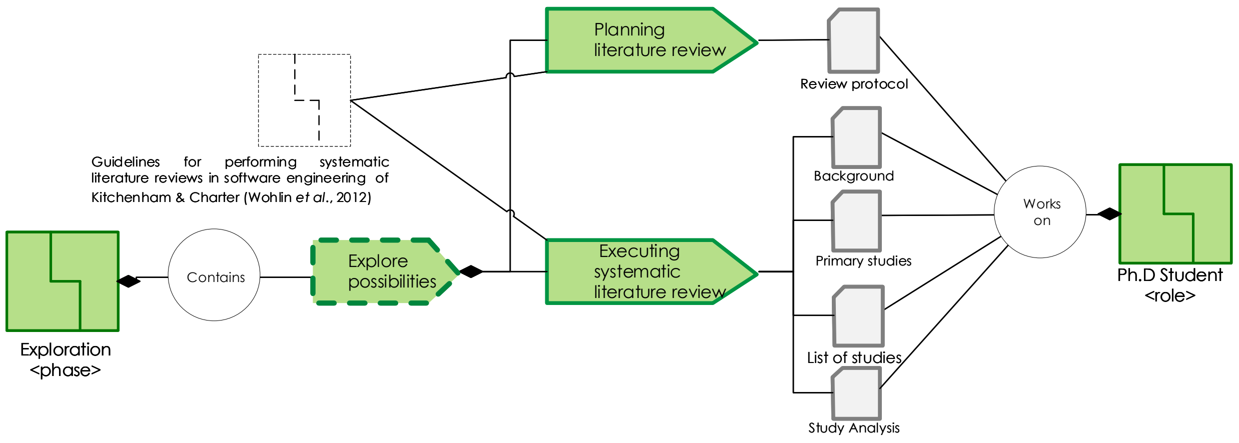

Planning literature review
Review protocol
Guidelines for performing systematic literature reviews in software engineering of Kitchenham & Charter (Wohlin et al., 2012)
Background
Works on
Ph.D Student <role>
Exploration <phase>
Contains
Explore possibilities
Executing systematic literature review
Primary studies
List of studies
Study Analysis

Figure 2-5 Exploration Phase. The Authors

2.3.2. Problem Formulation

A general problem and a set of specific problems are found based on the exploration phase. *Specifying problem statement*, *formulating research question*, and *formulating hypothesis* activities are performed for obtaining the *problem statement, research question*, and *hypothesis* and for defining the objectives (see Figure **2-6**).

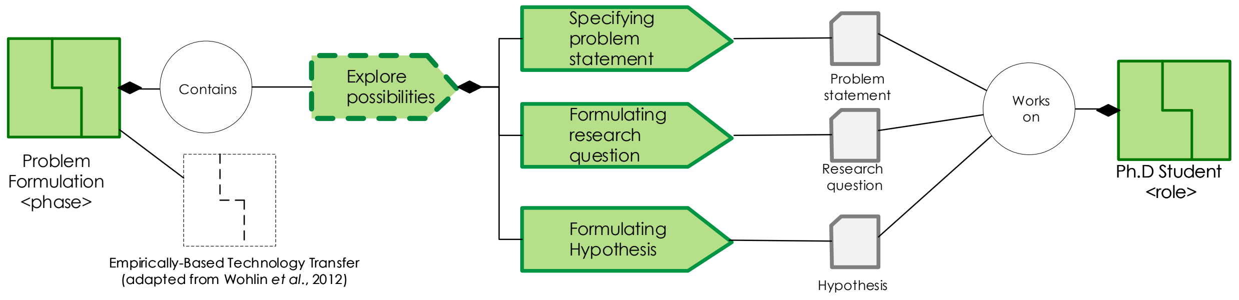

Specifying problem statement
Problem statement
Contains
Explore possibilities
Formulating research question
Research question
Works on
Ph.D Student <role>
Problem Formulation <phase>
Formulating Hypothesis
Hypothesis
Empirically-Based Technology Transfer (adapted from Wohlin et al., 2012)

Figure 2-6 Problem Formulation Phase. The Authors

2.3.3. Solution

A solution is proposed for refining event representation and mathematical notation in an extension to pre-conceptual schemas in SSD. *Characterizing events*, *defining linguistic, mathematical*, and *graphical structures*, and *including extension to PCS* are activities executed for producing an *event report*, the *linguistic, mathematical, and graphical structures*, and the *extension to PCS* (see Figure **2-7**).

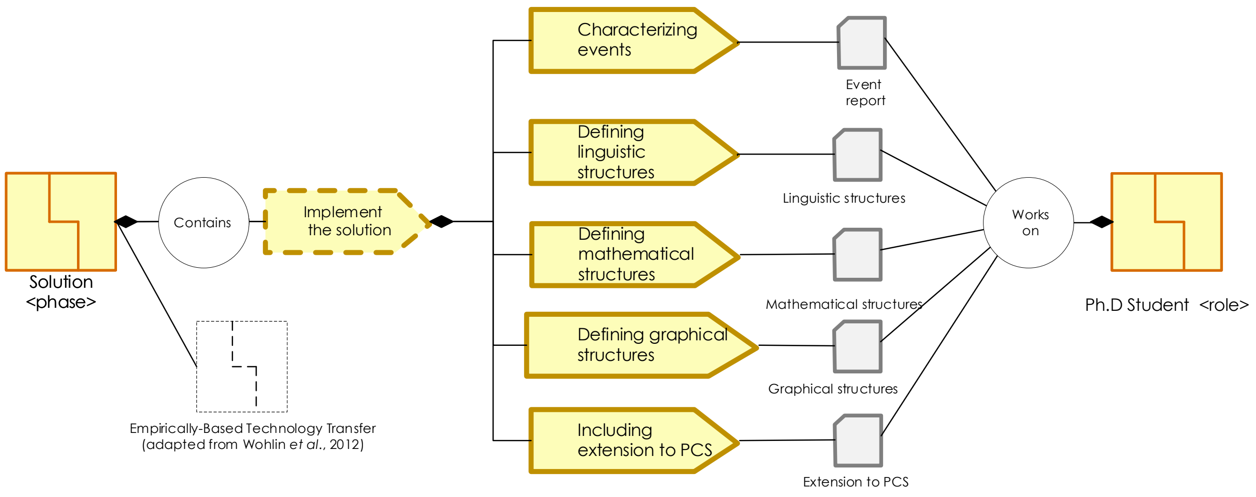

Solution
<phase>
Contains
Implement the solution
Empirically-Based Technology Transfer (adapted from Wohlin et al., 2012)
Characterizing events
Defining linguistic structures
Defining mathematical structures
Defining graphical structures
Including extension to PCS
Event report
Linguistic structures
Mathematical structures
Graphical structures
Extension to PCS
Works on
Ph.D Student <role>

Figure 2-7 Solution Phase. The Authors

2.3.4. Validation

An experiment is applied to several contexts for evaluating the amount of understanding of the proposed structures in the extended PCS. *Planning experiment*, *executing experiment*, *experiment data*, and *analyzing experiment* are activities developed for producing an *experiment design*, *experiment data*, and *experiment report* (see Figure **2-8**). Scientific papers are also produced in the different phases.

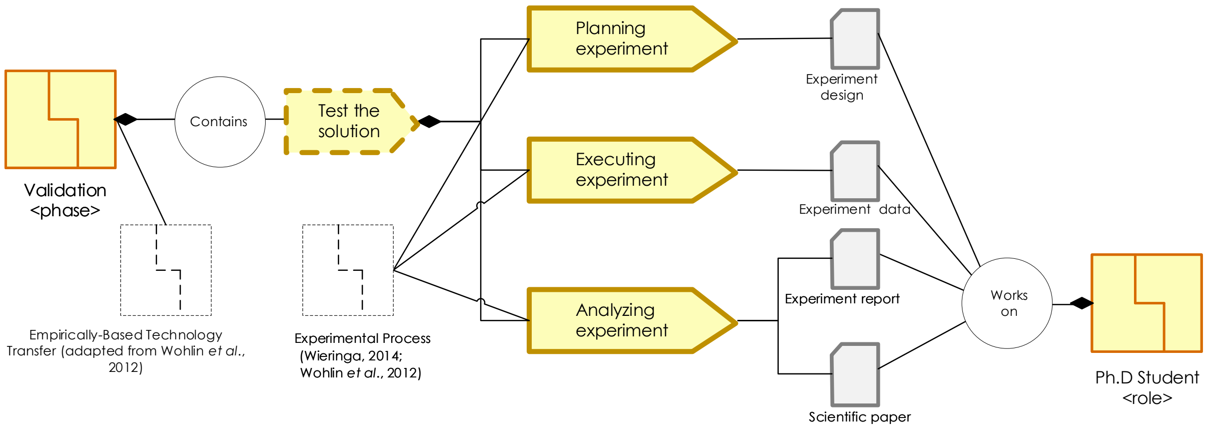


Figure 2-8 Validation Phase. The Authors

3 Research Problem

It is often interesting, in retrospect, for considering the causes that led to great events.
— Patricia Moyes

3.1. Motivation

Applicability of events in the development of software systems is an increasing trend (Campos-Rebelo *et al.*, 2015; Luckham, 2011). According to Luckham (2002) *"there is a fundamental reason for this broad applicability. It is simply because information systems are all driven by events."* Such applicability allows for developing complex systems (Campos-Rebelo *et al.*, 2015) in scientific domains (Kelly, 2015), *e.g.*, software for studying earthquakes and other natural events, medical software for detecting heart attacks, cancer, and other diseases, sensor system for ecological impact in environmental noise, pollution, air quality, climatic changes, etc., simulation software for detecting failures in chemical mixtures in a construction, and other automated and computer systems.

Event-driven trends have emerged from 1960 until today with a future perspective for event application (Luckham, 2011), as we show in Figure **3-1**.

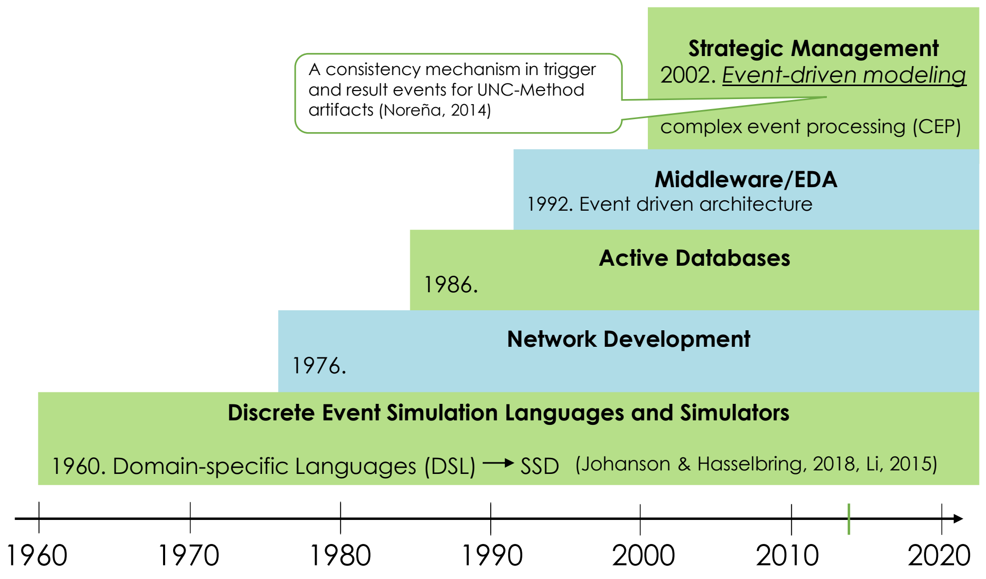

A consistency mechanism in trigger and result events for UNC-Method artifacts (Noreña, 2014)
Strategic Management
2002. Event-driven modeling
complex event processing (CEP)
Middleware/EDA
1992. Event driven architecture
Active Databases
1986.
Network Development
1976.
Discrete Event Simulation Languages and Simulators
1960. Domain-specific Languages (DSL) → SSD (Johanson & Hasselbring, 2018, Li, 2015)
1960
1970
1980
1990
2000
2010
2020

Figure 3-1 Event-driven Trends. The Authors based on Luckham (2011)

Events are used in: *discrete event simulation* for predicting the behavior of a system by using simulators and languages, *e.g.*, domain-specific languages (DSL) for programming a particular domain (Johanson & Hasselbring, 2018; Li, 2015), simulators with input and output events; *network development* for establishing communications (interoperability) and messages between systems (Noreña *et al.*, 2017); *active databases* for evaluating conditions when a new data arrives, *e.g.*, *on* event *if* Boolean-condition *then* action; *middleware* for communicating and transmitting messages; *event-driven architecture* (EDA) for developing publish/subscribe applications, receiving, and publishing events among interface services (Noreña & Zapata, 2018b); *strategic management*, *i.e.*, event-driven modeling, business intelligence (events trigger processes), and complex event processing (CEP, a set of techniques and tools for detecting events in real-time and reacting to them, which are also related to data; Luckham, 2002, 2011). This Ph.D. Thesis is motivated by this trend of applicability of events.

Noreña (2014) propose *a consistency mechanism in the trigger and result events for UNC-Method artifacts*, M.Sc. Thesis from the *event-driven modeling* trend. UNC-Method is a software development method of the *Universidad Nacional de Colombia* (Zapata, 2012). This M.Sc. Thesis is proposed for generating consistency in the events from the artifacts of such a method, *i.e.*, controlled dialogue, elicitation cards, pre-conceptual schema, process diagram, process diagram explanatory table, event interaction graph, and state machine diagram. Our Ph.D. Thesis is promoted and motivated by such a proposal from the continuity in the event work, especially in the analysis of verbs related to eventual relationships and event representation.

3.2. State of the Art

3.2.1. Planning Literature Review

Review protocol includes the *study criteria*, which are presented in the Table **3-1** for developing the systematic literature review (Wohlin *et al.*, 2012). A primary study (Haisjackl *et al.*, 2018) allows for defining our *research questions* (*RQ*) to be used in the literature review. According to Haisjackl *et al.* (2018), syntactic errors are generally identified in the processes while other problems remain unattended, *i.e.*, syntactic errors related to events. Also, they propose an exploration of other challenges in modeling notation when a process is created. Consequently, we suggest the questions *RQ1* and *RQ2*. After, we find mathematical models in scientific modeling for representing events (Mezerins, 2014; Sarno *et al.*, 2015), and then we propose the questions *RQ3* to *RQ7*.

Conforming to the *study criteria* (see Table **3-1**) the reviewed studies are grouped into four approach categories (list of studies and study analysis): software modeling, scientific modeling, software and scientific modeling, and PCS approaches. Finally, we identify approaches including linguistic structures for events from such categories.

Table 3-1 Study criteria. The Authors

Inclusion criteria	Search criteria	(i) Approach types—diagram (D), framework (F), graph (G), method (Me), model (M), and tool (T) (ii) Event-driven modeling (iii) Software modeling approaches including event representation (iv) Scientific modeling approaches including event representation (vi) PCS approaches related to events and mathematical notation
	Search sources	ACM (especially, ACM International Conference on Distributed Event-Based Systems), IEEE Explore, Scient Direct, Springer Books, Springer Links, Scopus, Google Scholar, OMG webpage
	Keywords	"event," "event driven," "event-driven," "event based," "event-based," "event modeling," "event modelling," "event representation," "software modelling," "software & modeling," "business process," "scientific software," "scientific application,", "science software," "scientific software domain," "engineering software for science," and "event simulation"
	Literature	Paper, chapter, book, thesis, and technical document
Exclusion criteria		(i) Software modeling approaches without event representation (ii) Scientific modeling approaches without event representation (iii) Methods, heuristic rules, and languages of programming (e.g., DSL) and testing based on models or events
Research questions		RQ1. What structures are used for representing events in scientific and software modeling (view)? RQ2. What linguistic errors are detected in events from models? RQ3. What models include mathematical notation for representing events in scientific software domains? RQ4. What mathematical structures are used? RQ5. What models include the time by using events? RQ6. What is used the model for? RQ7. What models include event functionality?

3.2.2. Executing Systematic Literature Review

Software Modeling Approaches

Some event-driven modeling approaches from software engineering are used for representing the behavioral view of a system. Business process model notation (BPMN) and unified modeling language (UML) are the most used notations. BPMN process model (Haisjackl *et al.*, 2018; OMG, 2014a) is used for representing some event types—none, timer, message, conditional, signal, error, etc. Time (*two weeks, one week*) and message events (*hold book, decline hold*) are shown in Figure **3-2**. UML activity (OMG, 2011), state machine (Chonoles, 2017), and sequence diagrams (OMG, 2015) are only used for representing none or statement events.

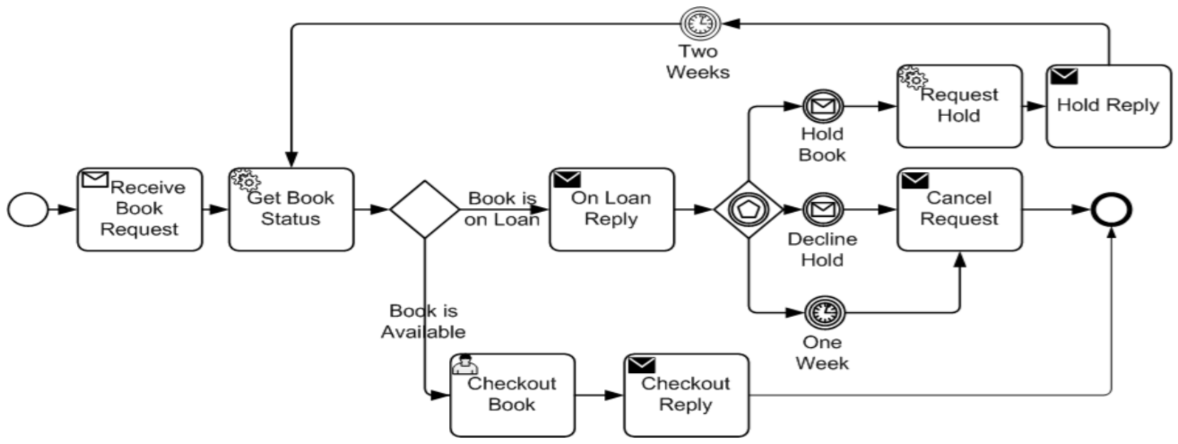


Figure 3-2 BPMN process model (OMG, 2014a)

UML state machine diagram includes the event notation *trigger event [guard condition] action* in a transition—connection arrow between states and activities, see such events (*[No Reserve], BorrowRequest[isCircBook]*, etc.) in a system of book reservation in Figure **3-3**. Event-driven process chain (EPC) diagram (Amjad *et al.*, 2017; Xia *et al.*, 2014; Xue *et al.*, 2013) and event interaction graph (EIG) in the UNC-Mehod (Zapata *et al.*, 2014) are used for representing the flow among events and processes. Events (*new repair task arrived, if no engineers are free*) in an EPC for a system of requests to engineers are shown in Figure **3-4**. Notation of the system modeling language (SysML) state machine diagram (Baouya *et al.*, 2015) includes conditional events, *e.g.*, *[sunny=true]*.

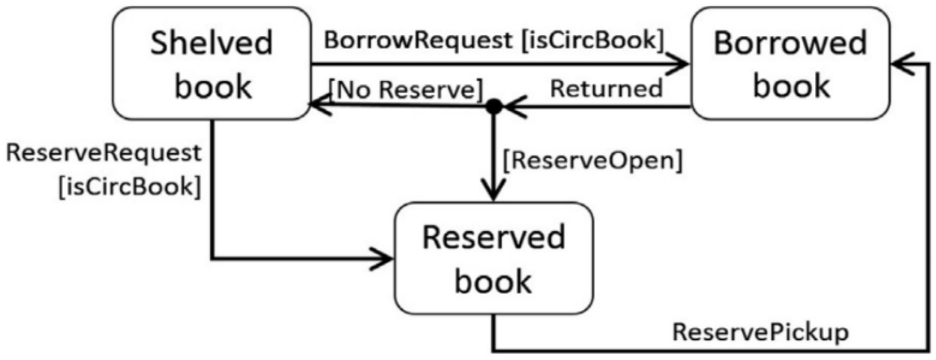


Figure 3-3 UML state machine diagram (OMG, 2014b)

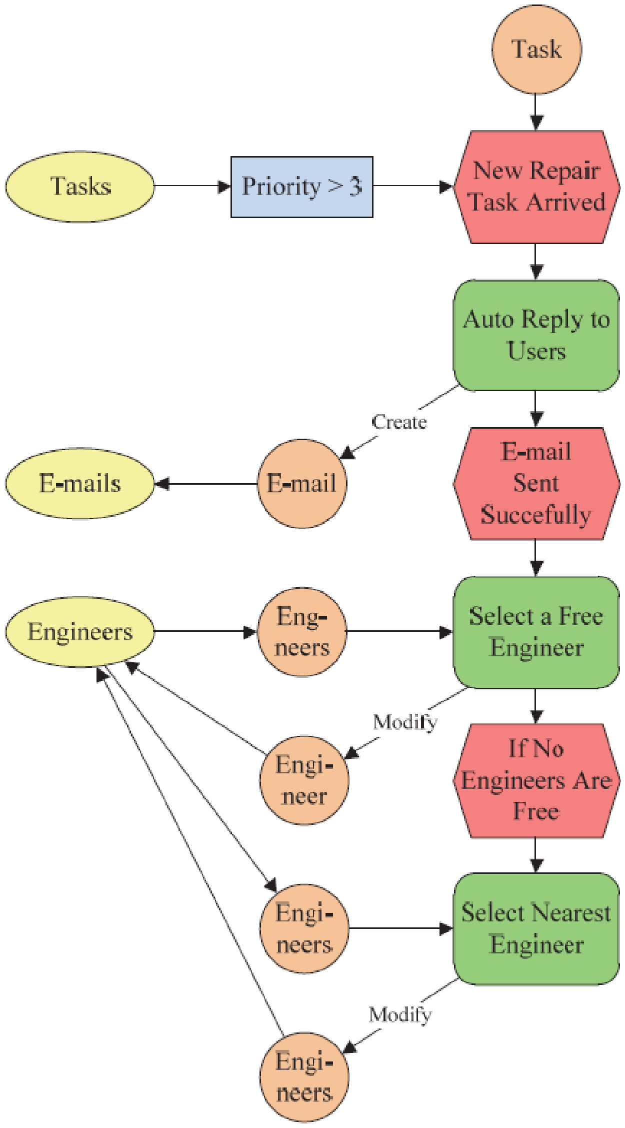
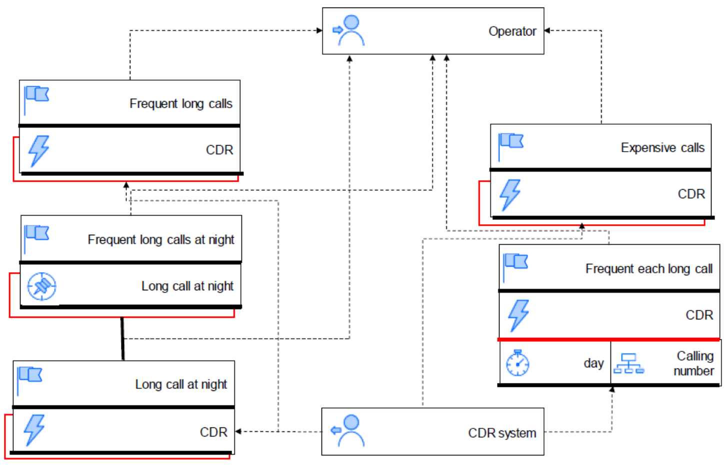


Figure 3-4 EPC (Xue *et al.*, 2013)

Some approaches are used for representing the structural view of a system. The event model (TEM, see Figure **3-5**) is used for modeling event-driven applications targeted to business users. TEM event logic/functionality is expressed by using TEM tables, whose detected event is *long call at night* in a system for detecting mobile phone fraud (Etzion *et al.*, 2016). Comprehensive event ontology (CEVO) is designed for recognizing and equating relationships from both textual data sources and knowledge bases (Shekarpour *et al.*, 2019), as we show in Figure **3-6**. CEVO has a hierarchy of communication, where several verbs are proposed for transfering message events, as we show in Figure **3-7**. Medit4CEP (Boubeta-Puig *et al.*, 2015, 2019) is a model-driven approach for CEP, which contains a tool for editing the model (see Figure **3-8**) with mathematical operators.



***Expensive calls* Logic**												
Row #	**When Expression**	**When Start**	**When End**	**Partition by**	**Filter on event**				**Pattern**		**Filter on pattern**	
				Calling number	other_party_tel_number <CDR>		call_direction <CDR>		SUM(total_call_charge_amount<CDR>)			
1	every 6 hours	first CDR		same	member of	premium services	=	1	>	100		

Figure 3-5 TEM model and TEM table for the mobile phone fraud (Etzion *et al.*, 2016)

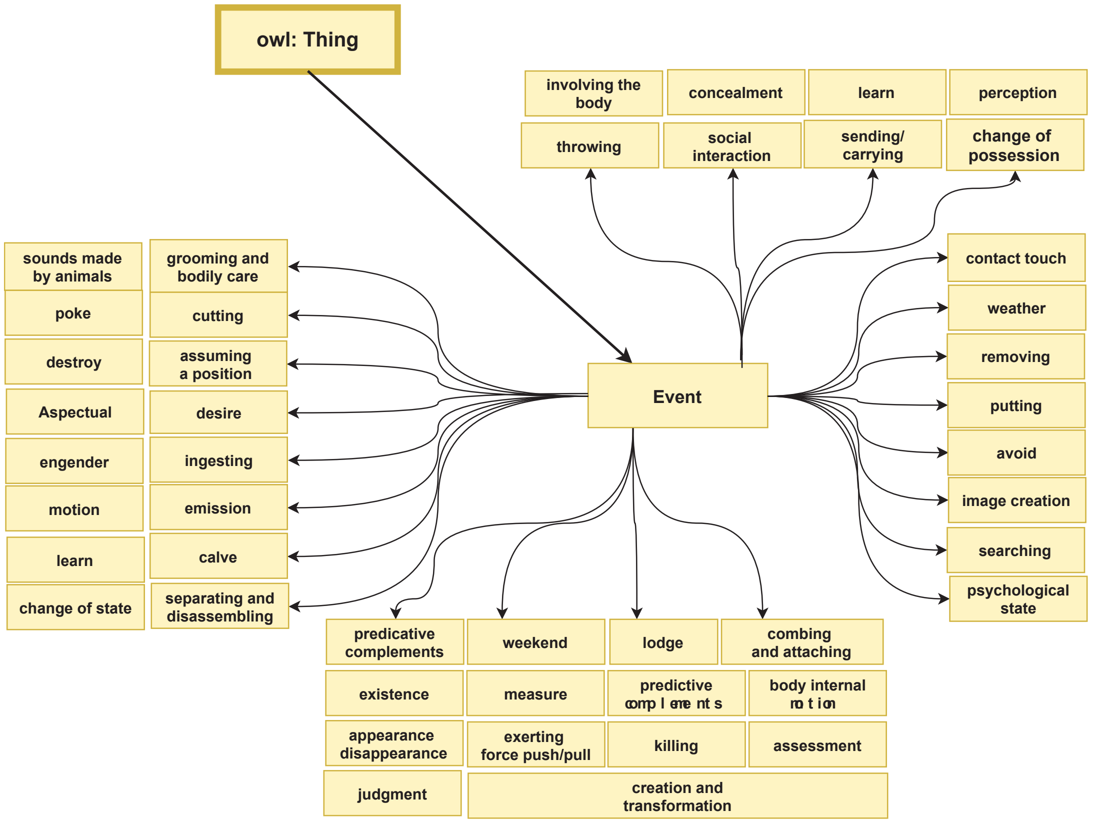


Figure 3-6 CEVO ontology (Shekarpour *et al.*, 2019)

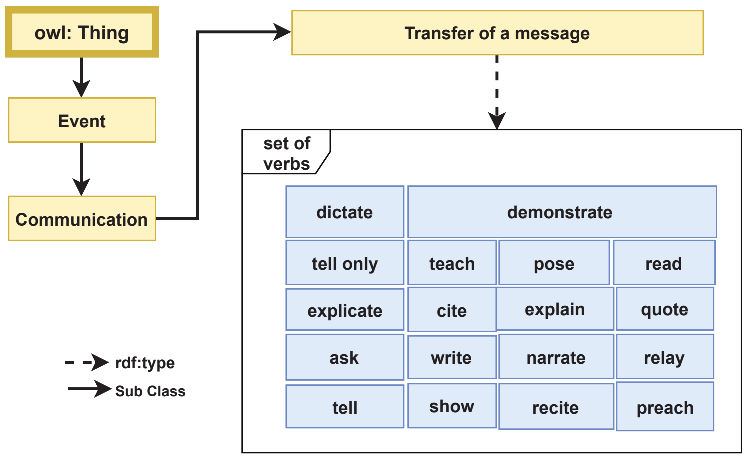


Figure 3-7 Verbs associated with message events in CEVO (Shekarpour *et al.*, 2019)

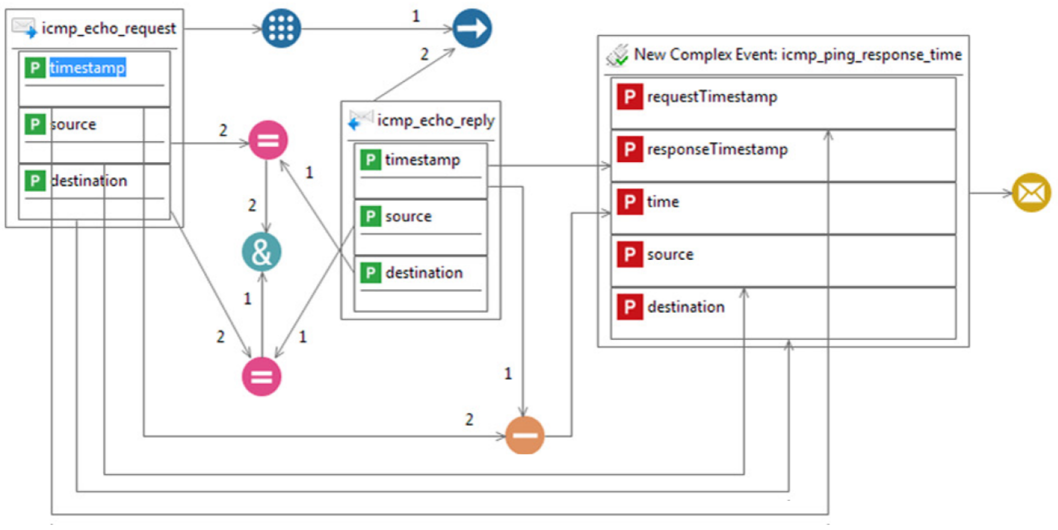


Figure 3-8 Medi4CEP diagram for representing complex events (Boubeta-Puig *et al.*, 2015)

Pre-conceptual schemas (PCS) are models used for modeling any domain (PCS are part of UNC-Method). PCS include notation for representing none/statement trigger events *e.g., file arises, user arises, image arises*; conditional, *e.g., description_right = admin AND description_right = files*; and basic mathematical operations (see Figure **3-9**) in a multiple view, which allows a complete representation of the domain (Zapata, 2012). EIG is based on the PCS notation (Noreña, 2014).

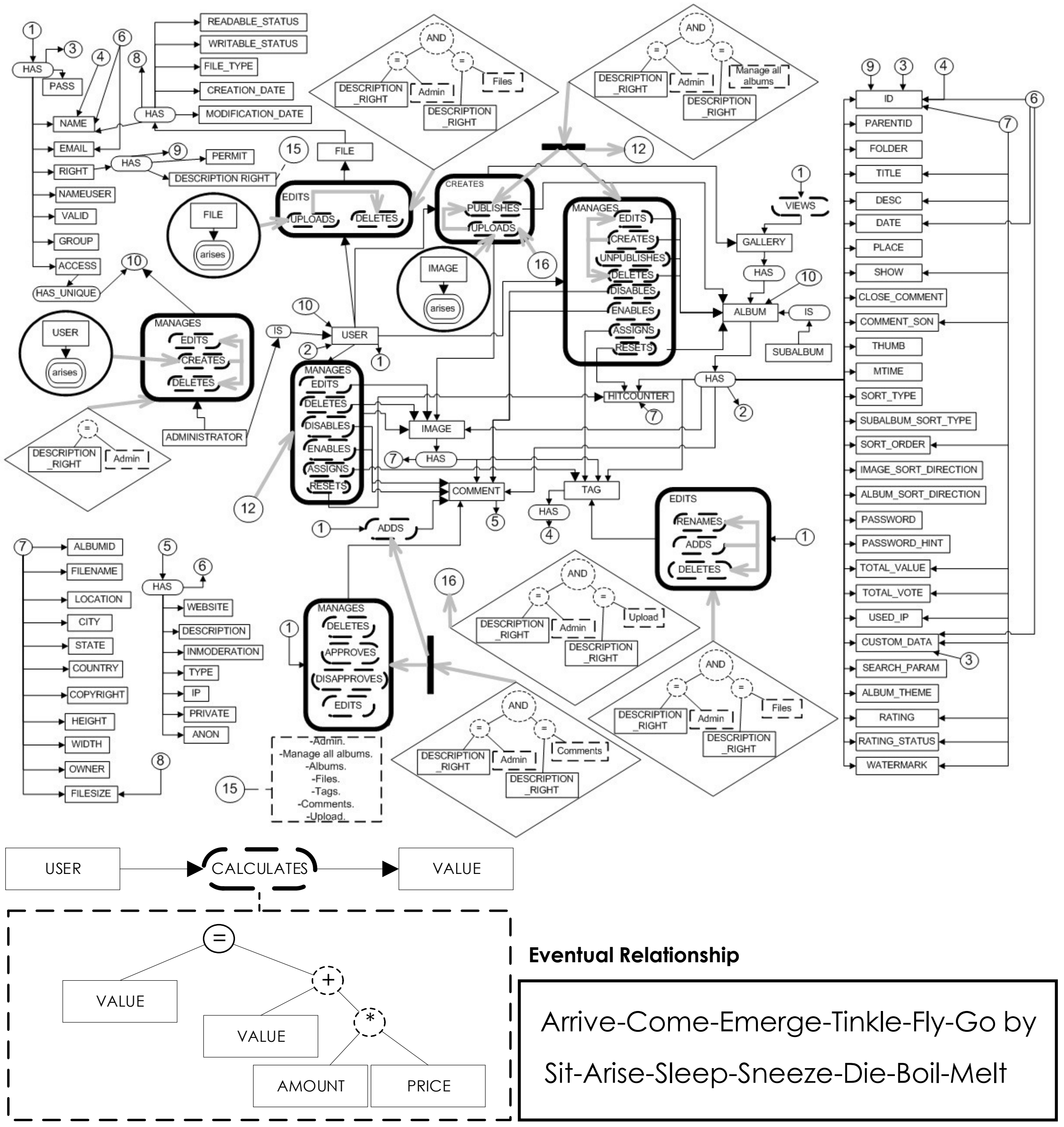


Figure 3-9 Pre-conceptual schema, mathematical operations, and eventual relationships (Zapata, 2012; Chaverra, 2011)

We synthesize the software modeling approaches (11) in Table **3-2** for partially answering the research questions (see Table **3-1**) of the literature review.

The answer to *RQ1* is the following: commonly, events are graphically and linguistically represented in the behavioral view of a system from software modeling (9 out of 11). All of the reviewed approaches use graphical and linguistic structures. BPMN process model, EPC, Medit4CEP, and PCS present mathematical structures (4 out of 11); however, BPMN process model and EPC lack a structural view. Medit4CEP and PCS are used for representing a multiple view of the system. Medit4CEP is only used in applications for complex event processing in event patterns and data generation (see Figure **3-8**) while PCS is used in applications of any domain (see Figure **3-9**).

Table 3-2 Software modeling approaches. The Authors

Authors	Approaches	Primary Studies	Software modeling			Usability			Events				Mathematical notation				Time from events
			Type	Structural view	Behavioral view	Domain knowledge	Software development	Simulation	Graphical structures	Linguistic structures	Mathematical structures	Functionality	Logical	Relational	Basic	Complex	
Haisjackl *et al.*, 2018 OMG, 2014a	BPMN	X	M		X	X	X	X	X	X	X		X				X
OMG, 2015 OMG, 2011	UML (Activity diagram	X	D		X	X	X		X	X			X	X			X
Chonoles, 2017 OMG, 2015	State Machine diagram	X	D		X	X	X		X	X		X	X	X			
OMG, 2015	Sequence diagram)	X	D		X	X	X		X	X							
Amjad *et al.*, 2017 Xia *et al.*, 2014 Xue *et al.*, 2013	Event-driven Process Chain (EPC)	X	D		X	X	X		X	X	X		X				
Zapata *et al.*, 2014	Event Interaction Graph	X	G		X	X	X		X	X							X
Shekarpour *et al.*, 2019	Comprehensive event ontology (CEVO)	X	M	X		X	X		X	X							X
Etzion *et al.*, 2016	The Event Model (TEM)		M	X			X		X	X		X					X
Bauoya *et al.*, 2015	SysML State Machine		D		X	X	X		X	X							
Boubeta-Puig *et al.*, 2015; 2019	Medit4CEP		T/M	X	X	X	X		X	X	X		X	X	X		X
Noreña, 2014 Zapata, 2012	Pre-conceptual Schemas (PCS)	X	M	X	X	X	X		X	X	X		X	X	X		

The answer to *RQ3* and *RQ4* is the following: some approaches include logical operators (6 out of 11) in the mathematical notation, but they lack other mathematical notation: relational (4 out of 11), basic (2 out of 11), and complex (2 out of 11) operations. Medit4CEP include logical, relational, and basic operators for relating conditions in complex events (see Figure **3-8**) while PCS include such operators in mathematical equations (see Figure **3-9**).

The answer to *RQ5* and *RQ7* is the following: most approaches lack representation of the time from events (5 out of 11), and event functionality (2 out of 11). UML state machine diagram includes the event funcionality by using the structure *trigger event [guard condition] action* and TEM includes the event functionality by using tables related to data events (see Figure **3-5**).

The answer to *RQ6* is the following: most approaches are used for representing the domain knowledge (10 out of 11) and the system in software development (11 out of 11), but they lack scientific components to be used in the development and simulation of scientific domains (1 out of 11).

Scientific Modeling Approaches

Some event-driven modeling approaches from science are used for analyzing phenomena/events by using the behavioral view of the system. Commonly, systems are discretely modeled for calculating output values in a discrete set of instants. A block diagram is used for representing an event-triggered controller system (see Figure **3-10**), which allow for representing the system, processes, and events by using concepts and mathematical notation, whose events are input and output data from sensors (Zhong & He, 2016).

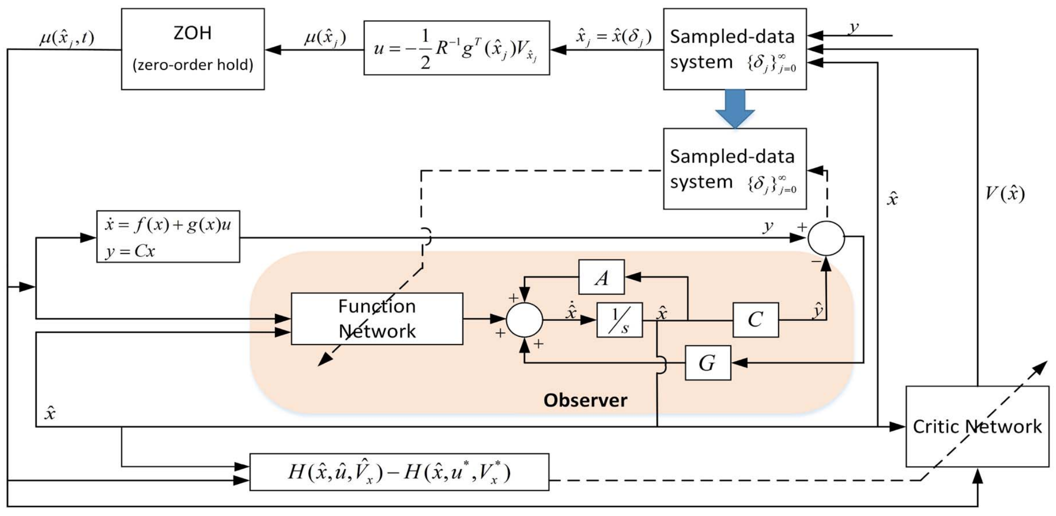

ZOH
(zero-order hold)
$\mu(\hat{x}_j, t)$
$\mu(\hat{x}_j)$
$u = -\frac{1}{2}R^{-1}g^T(\hat{x}_j)V_{\hat{x}_j}$
$\hat{x}_j = \hat{x}(\delta_j)$
Sampled-data system $\{\delta_j\}_{j=0}^{\infty}$
y
Sampled-data system $\{\delta_j\}_{j=0}^{\infty}$
$\hat{x}$
$V(\hat{x})$
$\dot{x} = f(x) + g(x)u$
$y = Cx$
Function Network
A
$\dot{\hat{x}}$
$1/s$
C
$\hat{y}$
G
Observer
Critic Network
$H(\hat{x}, \hat{u}, \hat{V}_x) - H(\hat{x}, u^*, V_x^*)$

Figure 3-10 Block diagram in event-triggered controller (Zhong & He, 2016)

A Petri net is used for partially controlling discrete event systems (see Figure **3-11**; Petri net is used for representing the state changes in the system and the block diagram is used for representing the state-feedback control system for Petri nets). An example of a maze as the discrete event system with sensors and actuators is carried out (Luo & Zhou, 2016). In this example, a cat and a mouse are in the maze, and the system objective is preventing the cat from eating the mouse by controlling the gates for being opened or closed when the cat and mouse change room (events). A Petri net is used for simulating processes formed by conditionals (AND parallel, OR), which allow for discovering relationships contained in event logs (Sarno *et al.*, 2015).

A finite automaton (a model based on mathematical foundations for analyzing state transitions) is used for controlling networks by obtaining deterministic outputs from initial states/inputs (Zhang & Zhang, 2016). A finite state machine (FSM, a model for representing state transitions) is used for modeling pattern matching queries for scalable complex event processing (Balkesen *et al.*, 2013). A Markov model (a discretized mathematical model for event occurrence probability) is used for estimating the event-triggered sensor data, which is applied to a monitoring system in the manufacturing industry for detecting failures (events) in a soft-drink filling machine during routine operation (Chen *et al.*, 2017).

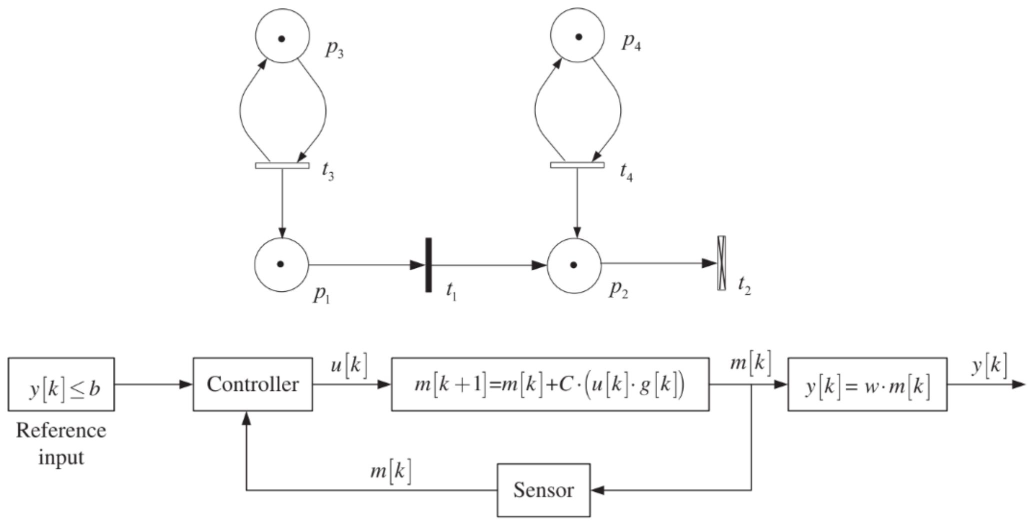


Figure 3-11 Petri net and block diagram (Luo & Zhou, 2016)

Markov model in Figure **3-12 (a)** is used for modeling the randomness of actuator failures in control systems (Wang *et al.*, 2017). Event-triggered control in Figure **3-12 (b)** is used for controlling the states of a system by using sensor-controller communication constraints (Xue & El-Farra, 2016). Pollution event model in Figure **3-12 (c)** is used for monitoring the dangerous and harmful chemical emissions in the enterprises and city infrastructure (Koltsov *et al.*, 2018). Event timer model in Figure **3-12 (d)** is used for representing signals in the digital domain, based on timing (Sudars *et al.*, 2015). Event timer model in Figure **3-12 (e)** is used in experimental studies for increasing the performance of an analog signal digitizing hardware (Event Timer A033-ET; Mezerins, 2014).

(a) $\|e_e(t)\| + \beta_o(1+\epsilon) \quad \leq \quad [\epsilon - \alpha_o(\epsilon+1)]\|\eta(t)\|$

for all $t \geq 0$, *where:*

$$\epsilon = \frac{\alpha_\omega - 4\Delta_A\|P\| - 4\Delta_B\|P\|\|K\|}{4(\|P\widehat{B}K\| + \Delta_B\|P\|\|K\|)}$$

(b) $[y((m_k+n)h) - y(m_kh)]^T \Lambda(\alpha(t))[y((m_k+n)h) - y(m_kh)] \leqslant \lambda^2(\alpha(t))\, y^T(m_kh)\,\Lambda(\alpha(t))\, y(m_kh)$

(c) $$\frac{dc}{dt} = V(KcV) - V(uc) + Q^C + R^C$$

(d) $x_k = A_r\cos[2\pi f_r(\tau_k - t_k)]$

(e) $x(t_k) = A_r\sin(2\pi f_r\hat{t}_k + \varphi_r)$

Figure 3-12 Mathematical models for representing events, **(a)** Markov model to reliable event-triggered (Wang *et al.*, 2017), **(b)** ETC (Xue & El-Farra, 2016), **(c)** Pollution event model (Koltsov *et al.*, 2018), **(d)** and **(e)** Event timer (Sudars *et al.*, 2015; Mezerins, 2014)

A Timing-idea graph is used for analyzing time event patterns (Wang *et al.*, 2016). A causal network is a graphical model used for designing complex representations of mental states by using sensors (Treur, 2016). Bayesian networks (graphical models, which contain mathematical models and algorithms in a separate way) are used for predicting complex events by using two dimensions: event type and time when new data arrives (Wang *et al.*, 2018). Neural networks are used for predicting clinical events (medical conditions or diagnosis of a patient) in electronic health records (Choi *et al.*, 2016). Event-based hybrid state estimation is an mathematical model for estimating states in the stochastic hybrid system, *e.g.*, sensors only transmit their measurements to an estimator when predefined events happen (Lee & Hwang, 2015).

Some event-driven modeling approaches from science are used for analyzing events by using the structural view of the system. Ontology-based vaccine and drug adverse event (acute and chronic thyroiditis, influenza vaccine, etc.) representation is an approach for relating entities and concepts in a specific biomedical domain. It also has conditions in the reactions *i.e.*, fever >= 10 %, redness >= 20 %, etc. (He, 2016). An ontology pattern for emergency event modeling is used for reusing existing emergency terminology. Two applied examples are: an event ontology of air pollution (see Figure **3-13**), which causes other events as death of residents, nausea, cough, etc; and an event ontology of water pollution (see Figure **3-14**) caused by vehicle chemical leakage, producing diarrhea, nausea, etc. (Liu *et al.*, 2016). Process-oriented event model (PoEM) ontology is used for relating real-world entities and their properties and detecting events by interpreting of their instantaneous status, *e.g.*, pump and well failure events in oil and gas industry (Patri *et al.*, 2014).

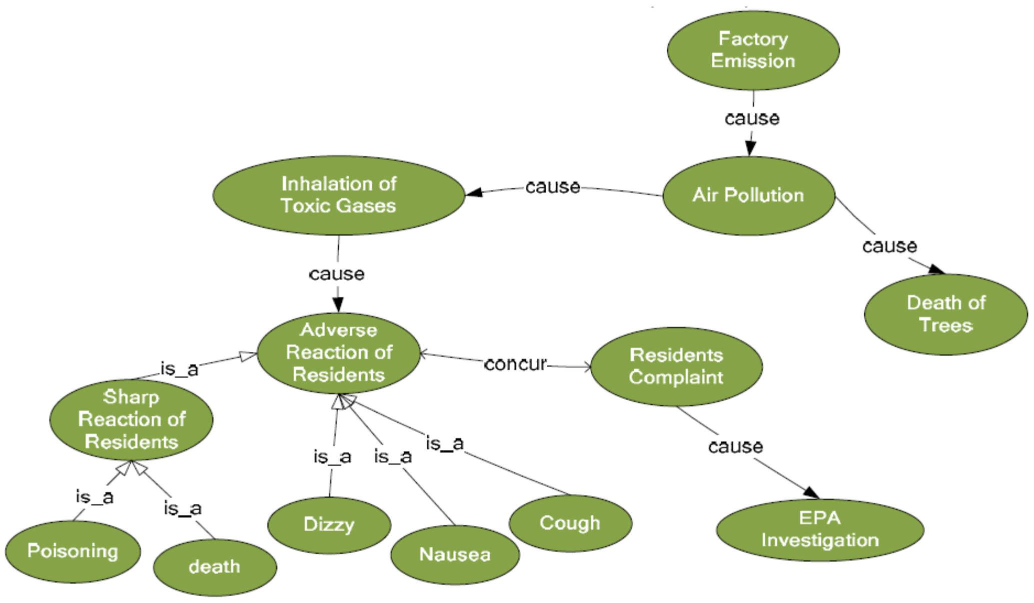


Figure 3-13 Event Ontology of Air Pollution (Zhang *et al.*, 2015)

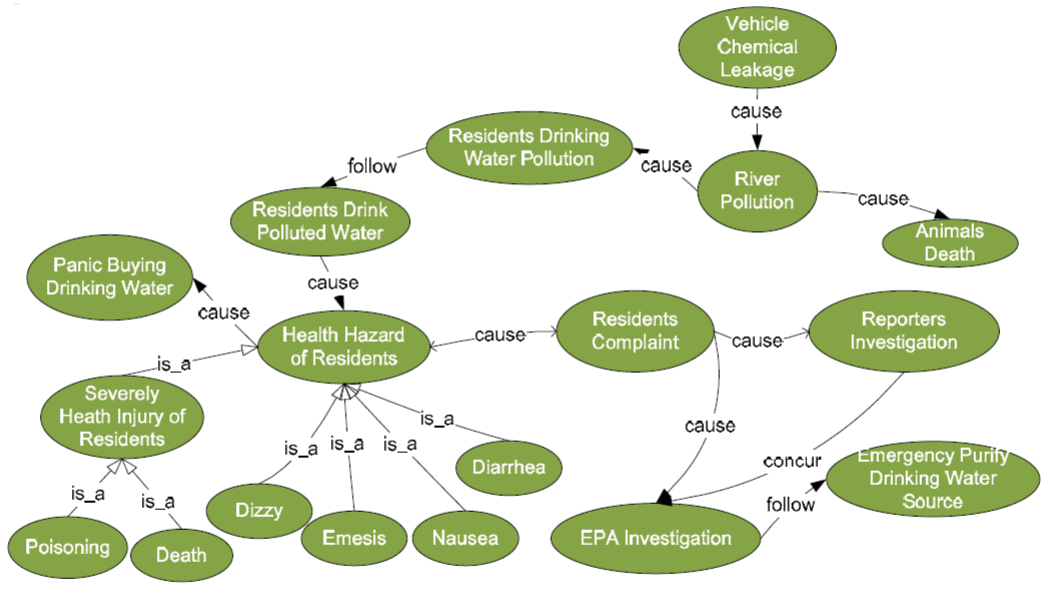


Figure 3-14 Event ontology of water pollution (Zhang *et al.*, 2015)

An event ontology based on the simple knowledge organization system (SKOS, a data model for sharing and linking knowledge organization systems via the Web) is shown in Figure **3-15**. Such an ontology is used for capturing the event-based knowledge by using static (place, language, objects) and dynamic aspects of an application domain (action and status) elements (Zhang *et al.*, 2015).

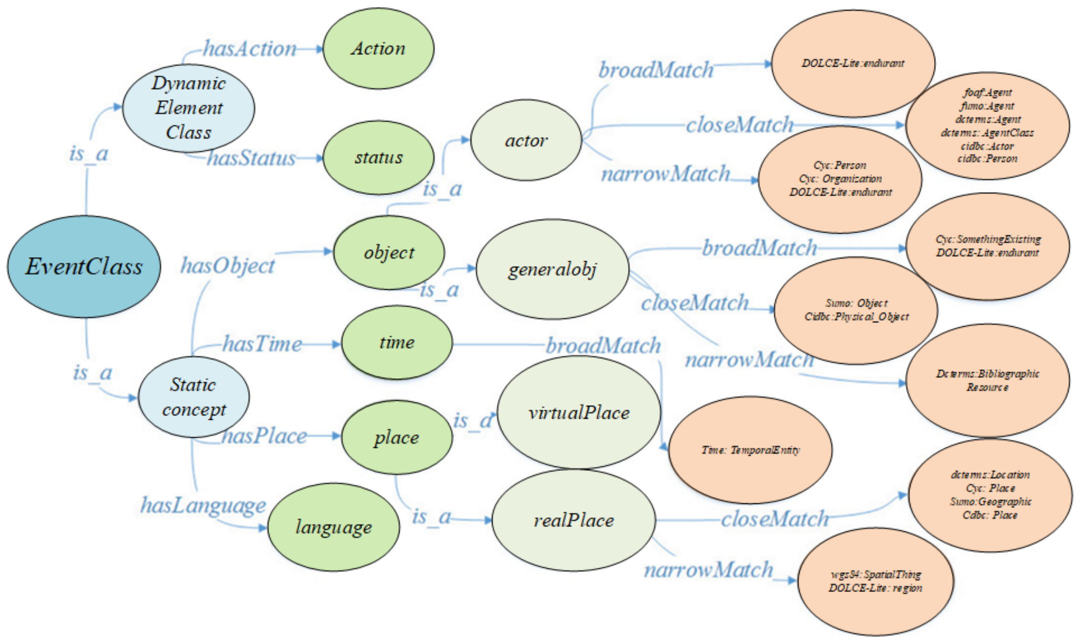


Figure 3-15 Event ontology based on SKOS (Zhang *et al.*, 2015)

Quadruple (contexts, events, relationships, rules) anonymity trajectory (QAT) ontology is used for representing contexts with location information about trajectory, geographical environment, etc. Such an ontology is shown in Figure **3-16**, including static (environment, *i.e.*, physical, meteorology) and dynamic aspects (status, *i.e.*, move and stop, action) for query events, and the semantic role *agent* for the actor (Zhu, 2018).

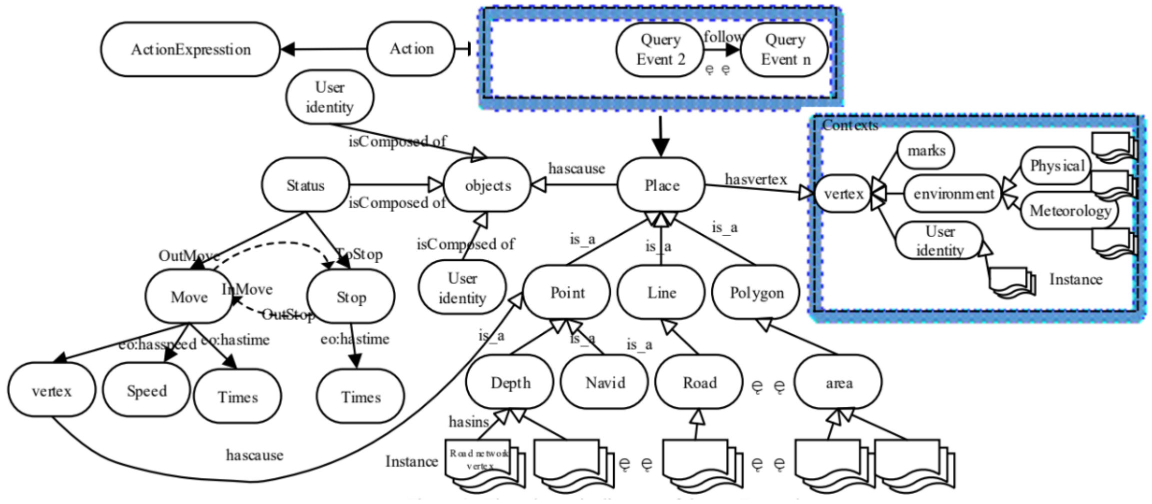


Figure 3-16 QAT ontology (Zhu, 2018)

We summarize the scientific modeling approaches (19) in Table **3-3** for partially answering the research questions (see Table **3-1**) of the literature review.

The answer to *RQ1* is the following: commonly, events are mathematically represented in the behavioral view of a system in scientific modeling (16 out of 19). Several mathematical structures are used (13 out of 18), but the graphical (8 out of 18) and linguistic structures are also used (5 out of 19). A block diagram is a graphical approach widely used where mathematical structures are represented in the same diagram (see Figures **3-10**). However, such a diagram also lacks components of a structural view of software modeling. Event ontology based on SKOS (Zhang *et al.*, 2015) and QAT ontology (Zhu, 2018) are structural approaches, which include dynamic elements as *status* and *action* (see Figure **3-16**). However, they lack components as processes and event functionality from a behavioral view.

The answer to *RQ3* and *RQ4* is the following: most approaches include complex mathematical notation (14 out of 19) and scientific concepts.

The answer to *RQ5* is the following: several approaches include the time from events (11 out of 19).

Table 3-3 Scientific modeling approaches. The Authors

Authors	Approaches	Primary Studies	Scientific modeling			Usability			Events				Mathematical notation				Time from events
			Type	Structural view	Behavioral view	Domain knowledge	Software development	simulation	Graphical structures	Linguistic structures	Mathematical structures	Functionality	Logical	Relational	Basic	Complex	
Zhong & He, 2016	Block diagram		D		X	X	X	X	X		X	X	X		X	X	X
Luo & Zhou, 2016 Sarno *et al.*, 2015	Petri Net	X	M		X	X	X	X	X		X	X			X	X	X
Zhang & Zhang, 2016	Finite Automaton		M		X	X		X	X		X	X			X	X	X
Balkesen, *et al.*, 2013	Finite State Machine		M		X	X	X	X	X		X	X				X	
Chen *et al.*, 2017 Wang *et al.*, 2017	Markov model		M		X			X	X		X	X				X	
Xue & El-Farra, 2016	Event-triggered control (ETC)		M		X			X			X	X			X	X	X
Kolsov *et al.*, 2018	Pollution event model		M		X			X			X	X		X	X	X	
Sudars *et al.*, 2015 Mezerins, 2014	Event Timing	X	M		X	X	X	X	X		X	X				X	X
Wang et al., 2016	Timing-Idea Graph		G		X	X	X	X	X		X				X		X
Treur, 2016	Causal network		M		X	X	X	X	X		X	X				X	
Wang *et al.*, 2018	Bayesian network		M		X	X	X	X	X		X	X			X	X	
Choi *et al.*, 2016	Neuronal network		M		X		X	X	X		X	X				X	
Lee & Hwang, 2015	Event-based hybrid state estimation		M		X		X	X			X	X				X	
He, 2016																	
Liu *et al.*, 2016																	
Patri *et al.*, 2014	Event Ontology		M	X	X	X	X	X	X	X							X
Zhang *et al.*, 2015					X												
Zhu, 2018																	

The answer to *RQ6* is the following: all of the reviewed approaches are used for simulating systems, representing the system in software development (15 out of 19), and domain knowledge (14 out 19), since commonly, algorithms and simulations are present in scientific modeling for analyzing a phenomenon.

The answer to *ARQ7* is the following: most approaches include the event funcionality in a mathematical model (14 out of 19, see Figure **3-12**).

Software and Scientific Modeling Approaches

A UML state machine diagram and a finite state machine (FSM) are used for representing the occurrence of an event in programs of logic controllers (Bazydlo *et al.*, 2014); such an approach is shown in Figure **3-17**. A BPMN process model and Petri nets are used for translating from process models to event structures, a formalism of behavioral relationships by expressing dependencies among events (Armas-Cervantes *et al.*, 2016). Such models are also used for formalizing event processing networks in simulators (Reinartz *et al.*, 2015). BPMN process models and a common information model (CIM) ontology are used for representing events (timer, message transactions) and chronology of tasks in the power system, a case study of energy scheduling business process in the indian power grid context (Ravikumar *et al.*, 2016). BPMN process models and decision requirement diagram (DRD) are used for representing diagnosis and treatment of patients affected by chronic obstructive pulmonary disease (see Figure **3-18**).

Such a disease is caused by smoking tobacco and exposing to polluted environments; the system is focused on monitoring and reducing the patient symptoms, whose severity determines which is the stage of the illness (Bazhenova *et al.*, 2019). UML class, sequence, and activity diagrams, annotations of the modeling and analysis real-time, and embedded systems (MARTE) are used in the UML/MARTE timeliness modeling method for describing time properties and constraints of the system, *i.e.*, a radar in the air traffic control center for detecting meteorological conditions (Xia *et al.*, 2019). A UML class diagram and finite automata are used for identifying event streams by using complex event patterns (Dávid *et al.*, 2018).

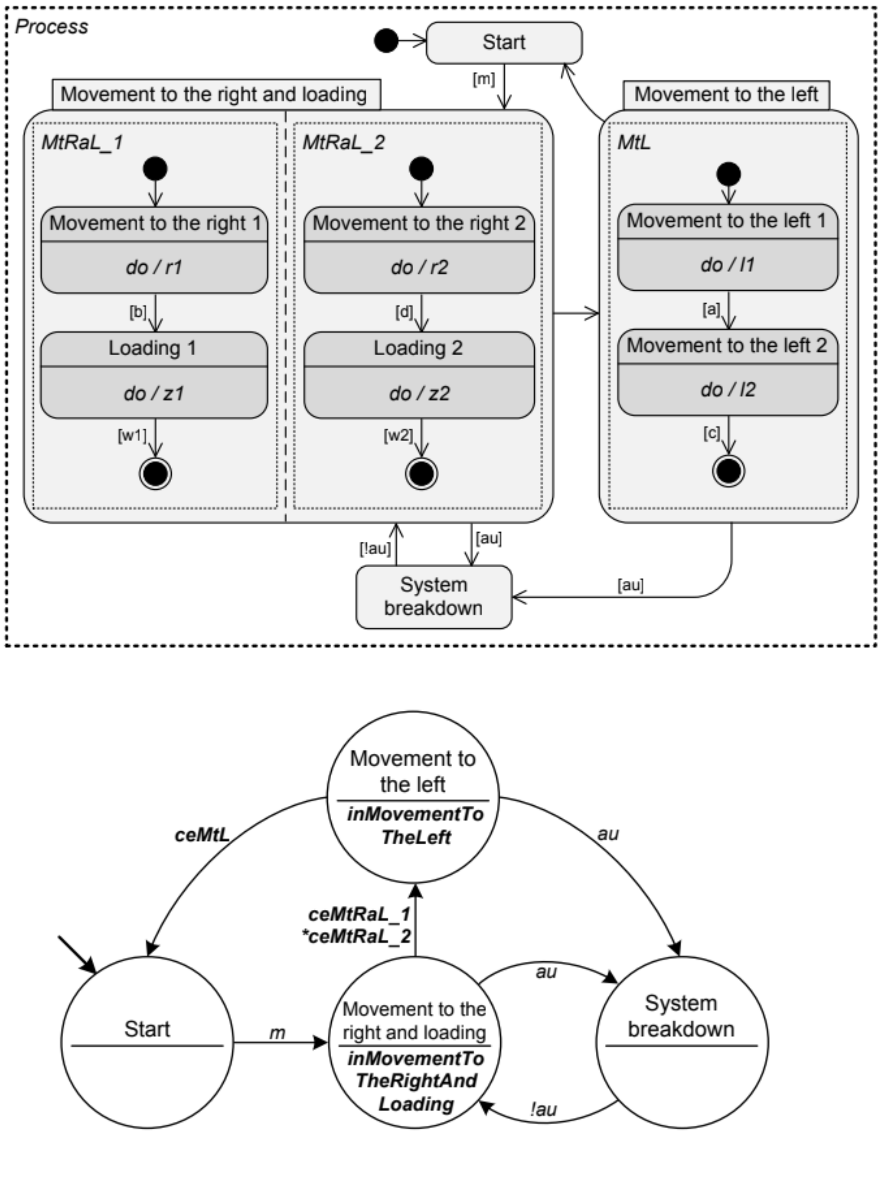


Figure 3-17 Translation of UML state machine diagram to FSM (Bazydlo *et al.*, 2014)

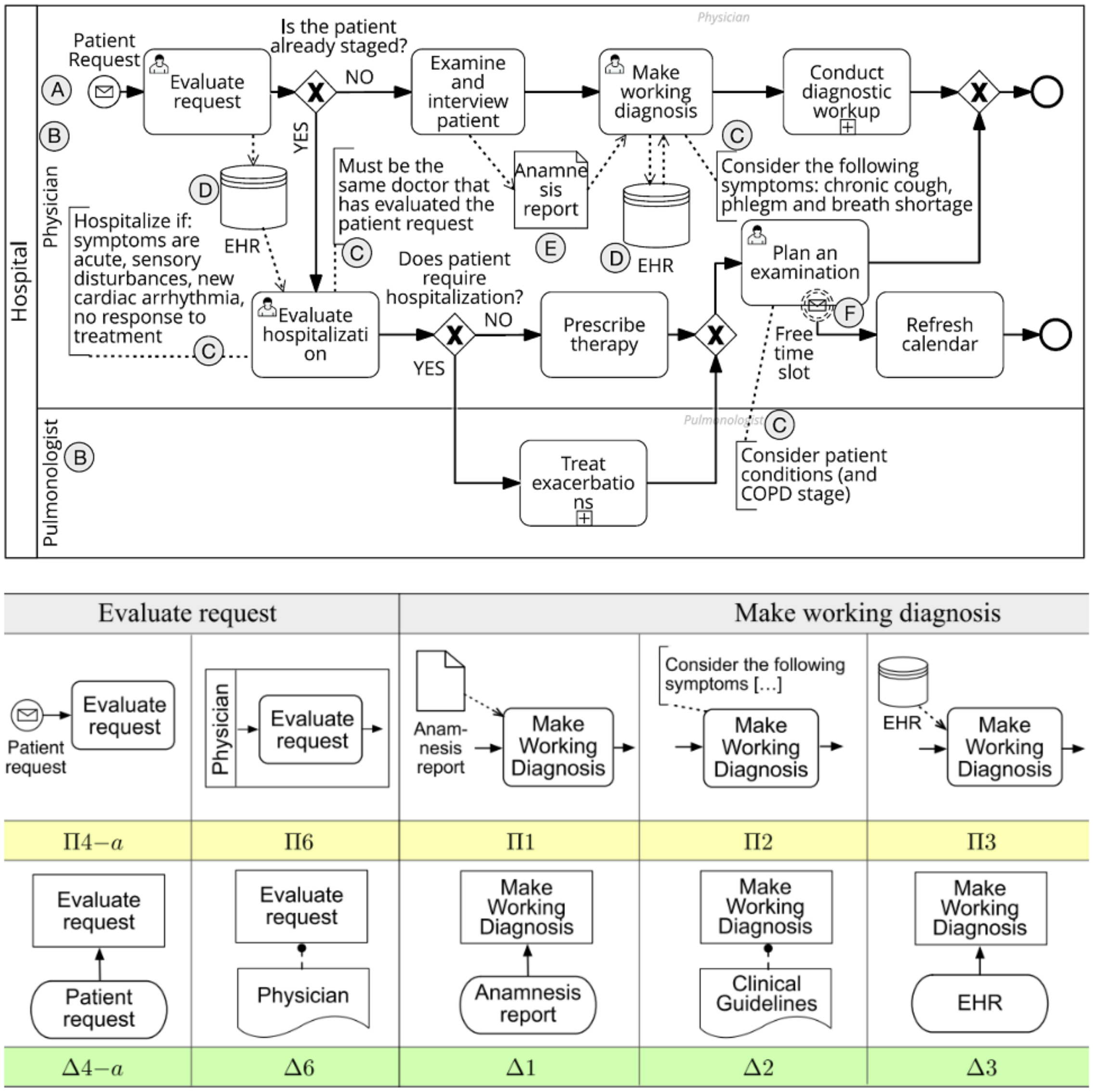
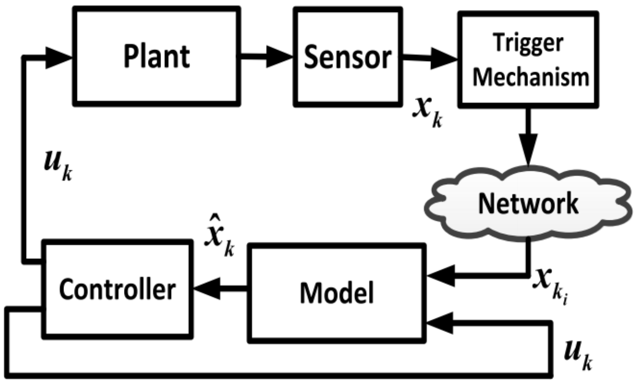

Hospital
Physician
Pulmonologist
Patient Request
Evaluate request
Is the patient already staged?
NO
YES
Examine and interview patient
Make working diagnosis
Conduct diagnostic workup
Must be the same doctor that has evaluated the patient request
Anamnesis report
Consider the following symptoms: chronic cough, phlegm and breath shortage
EHR
Hospitalize if: symptoms are acute, sensory disturbances, new cardiac arrhythmia, no response to treatment
Evaluate hospitalization
Does patient require hospitalization?
Prescribe therapy
Plan an examination
Free time slot
Refresh calendar
Treat exacerbations
Consider patient conditions (and COPD stage)
A
B
C
D
E
F
Evaluate request
Make working diagnosis
Patient request
Physician
Anam-nesis report
Make Working Diagnosis
Consider the following symptoms […]
EHR
Π4−a
Π6
Π1
Π2
Π3
Anamnesis report
Clinical Guidelines
Δ4−a
Δ6
Δ1
Δ2
Δ3

Figure 3-18 BPMN process model and DRD for diagnosing patients with chronic obstructive pulmonary disease (Bazhenova *et al.*, 2019)

An ontology and a syntax tree (a diagram with nodes and edges) in the event service model are used for generating event patterns and describing event service requests (Gao *et al.*, 2014). A block diagram, a flow diagram, and a mathematical model are used in the model based on event-triggered control (MBETC). Such a model is described in the context of reduced event sampled communication by using event-trigger conditions, as we show in Figure **3-19** (Sahoo *et al.*, 2015). Domain-specific requirements modeling for scientists (DRUMS) is a framework for describing requirements in the scientific domain and tool support (Li, 2015; Li *et al.*, 2015), as we show in Figure **3-20**.



$$e^s_{k+1} = x_{k+1} - \hat{x}_{k+1} = Ax_k + B\bar{F}(x_k)\bar{u}_k - A\hat{x}_k - B\hat{\bar{F}}(\hat{x}_k)\bar{u}_k$$
$$= Ae^s_k + B\tilde{\bar{F}}(x_k)\bar{u}_k + B(\hat{\bar{F}}(x_k) - \hat{\bar{F}}(\hat{x}_k))\bar{u}_k$$
$$k_i < k < k_{i+1}.$$

$$e^s_{k+1} = Ae^s_k + B\tilde{W}^T_k\Phi(\bar{x}_k)\bar{u}_k + B\Xi_k\bar{u}_k + B\hat{W}^T_k\tilde{\Theta}(\bar{x}_k, \hat{\bar{x}}_k)\bar{u}_k$$

$$e^s_{k+1} = B\tilde{W}^T_k\Phi(\bar{x}_k)\bar{u}_k + B\Xi_k\bar{u}_k, \quad k = k_i.$$

$$\hat{W}_k = \hat{W}_{k-1} + \frac{\gamma_k\alpha\Phi(\bar{x}_{k-1})\bar{u}_{k-1}e^{s^T}_k B}{1 + \|\Phi(\bar{x}_{k-1})\|^2\,\|\bar{u}_{k-1}\|^2} - \gamma_k\kappa\hat{W}_{k-1}$$
$$k_{i-1} \le k < k_i$$

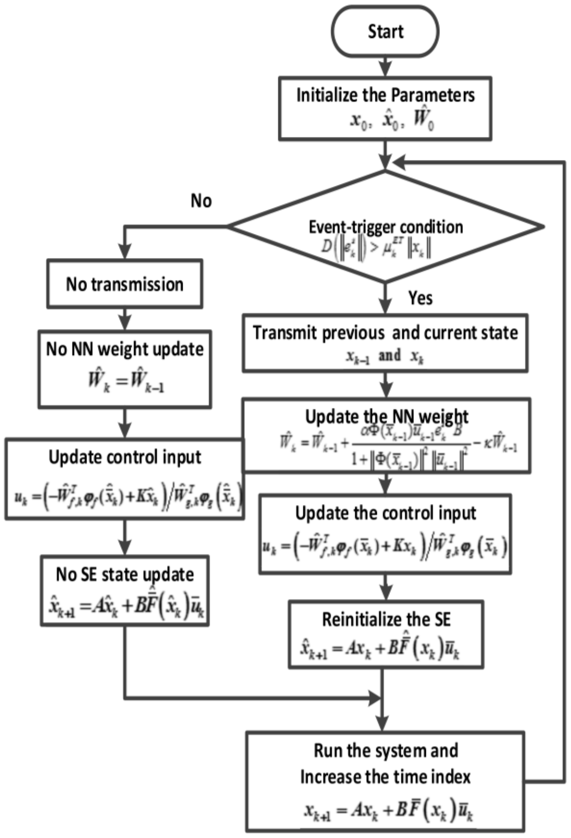

Start
Initialize the Parameters
$x_0,\ \hat{x}_0,\ \hat{W}_0$
Event-trigger condition
$D(\|e^s_k\|) > \mu^{ET}_k\|x_k\|$
No
Yes
No transmission
No NN weight update
$\hat{W}_k = \hat{W}_{k-1}$
Update control input
$u_k = \left(-\hat{W}^T_{f,k}\varphi_f(\hat{\bar{x}}_k) + K\hat{x}_k\right)/\hat{W}^T_{g,k}\varphi_g(\hat{\bar{x}}_k)$
No SE state update
$\hat{x}_{k+1} = A\hat{x}_k + B\hat{\bar{F}}(\hat{x}_k)\bar{u}_k$
Transmit previous and current state
x_{k-1} and x_k
Update the NN weight
$\hat{W}_k = \hat{W}_{k-1} + \frac{\alpha\Phi(\bar{x}_{k-1})\bar{u}_{k-1}e^s_k B}{1+\|\Phi(\bar{x}_{k-1})\|^2\|\bar{u}_{k-1}\|^2} - \kappa\hat{W}_{k-1}$
Update the control input
$u_k = \left(-\hat{W}^T_{f,k}\varphi_f(\bar{x}_k) + Kx_k\right)/\hat{W}^T_{g,k}\varphi_g(\bar{x}_k)$
Reinitialize the SE
$\hat{x}_{k+1} = Ax_k + B\hat{\bar{F}}(x_k)\bar{u}_k$
Run the system and Increase the time index
$x_{k+1} = Ax_k + B\bar{F}(x_k)\bar{u}_k$

Figure 3-19 MBETC (Sahoo *et al.*, 2015)

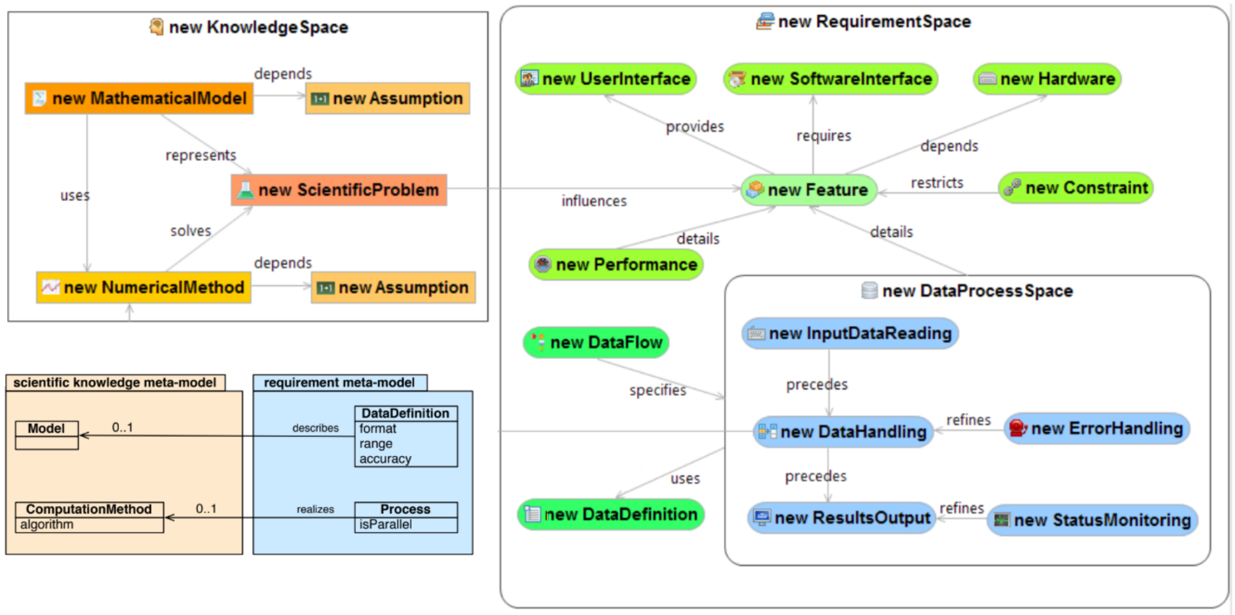


Figure 3-20 DRUMS (Li, 2015; Li *et al.*, 2015)

We synthesize the software and scientific modeling approaches used for representing events (10) in Table **3-4** according to the research questions (see Table **3-1**) of the literature review.

Table 3-4 Sofware and scientific modeling approaches. The Authors

Authors	Approaches	Primary Studies	Software and Scientific Modeling			Usability			Events				Mathematical notation				Time from events
			Type	Structural view	Behavioral view	Domain knowledge	Software development	Ssimulation	Graphical structures	Linguistic structures	Mathematical structures	Functionality	Logical	Relational	Basic	Complex	
Bazydlo, *et al.*,2014	UML state machine diagram/ Finite State Machine		**D**		X	X			X	X							
Armas-Cervantes *et al.*, 2016	BPMN/Petri Nets		**M**		X	X			X	X	X		X			X	X
Reinartz, *et al.*, 2015			**M**		X	X		X	X	X	X		X				
Ravikumar *et al.*, 2016	BPMN/Ontology	X	**M**	X	X	X	X	X	X	X	X		X				X
Bazhenova *et al.*, 2019	BPMN/Decision requirement diagram		**M**		X	X			X	X	X		X	X			
Xia *et al.*, 2019	UML/MARTE method		**Me**	X	X	X		X	X	X							X
Dávid *et al.*, 2018	UML class diagram/ Finite Automata		**D**	X	X	X	X	X	X	X	X	X	X				
Gao *et al.*, 2014	Ontology/ Syntax tree		**D**	X	X	X	X	X	X	X	X	X	X	X	X	X	
Sahoo *et al.*, 2015	MBETC (Model based on ETC) Flow diagram/block diagram/ mathematical model		**D/M**		X	X	X	X	X	X	X	X	X	X	X	X	X
Li, 2015 Li *et al.*, 2015	DRUMS (Domain-specific Requirements Modeling) for scientists		**F**	X		X	X									X (conceptually)	

The answer to *RQ1* is the following: commonly, events are represented by using graphical (9 out of 10), linguistic (9 out of 10), and mathematical (7 out of 10) structures in behavioral (9 out of 10) and structural (5 out of 10) views of a system in software in scientific modeling. BPMN/ontology (Ravikumar *et al.*, 2016), UML/MARTE method (Xia *et al.*, 2019), UML class diagram/finite automata (Dávid *et al.*, 2018), ontology/syntax tree (Gao *et al.*, 2014), and DRUMS (Li, 2015) present behavioral and structural views, but such views are separated into two and more models.

The answer to *RQ3* is the following: most approaches include mathematical notation (8 out of 10) and scientific concepts.

The answer to *RQ4* is the following: some approaches include logical (7 out of 10), relational (3 out of 10), basic (2 out of 10), and complex (4 out of 10) operators.

The answer to *RQ5* is the following: some approaches include time from events (4 out of 10).

The answer to *RQ6* is the following: most approaches are used for representing the domain knowledge (10 out of 10) and the system in software development (5 out of 10) and simulation (6 out of 10).

The answer to *RQ7* is the following: some approaches include the event functionality in a mathematical model (3 out of 10).

Such approaches are intended to address the gap between event-driven software modeling and event-driven scientific modeling in science and engineering projects; however, they lack a model for integrating scientific concepts/terminology consistently (without abbreviations, with complete names of concepts and variables) in order to understanding components of a domain in software engineering. The closest approach is DRUMS (Li, 2015), which is directly focused on requirements modeling for scientists from an architecture level (see Figure **3-20**); however, DRUMS lacks event representation and functionality (internal logic). The flow diagram in the MBETC model (Sahoo *et al.*, 2015) integrates into the same diagram a mathematical notation in events and processes in a behavioral view (see Figure **3-19**); however, such a model lacks concepts and relationships in a structural view of the system for analyzing the domain.

PCS Approaches

Pre-conceptual schemas are used for solving communication problems between analysts and stakeholders in software engineering (Zapata, 2007). Some generations of software engineers, which recognize schemas include structures for representing knowledge related to any domain (Zapata-Tamayo & Zapata-Jaramillo, 2018). Undergraduate, M.Sc., and Ph.D. students have proposed approaches by using PCS. However, we only focus on approaches including event representation and mathematical notation. Basic mathematical equations are proposed for specifying dynamic relationships (see Figure **3-9**) in order to automatically generate functional prototypes by using PCS (Chaverra, 2011).

UNC-Method is a problem-based software development method, which is focused on describing a domain knowledge for a future software system, which is generated as a solution to the domain problem (Zapata, 2012). The event representation in PCS is incorporated in UNC-Method for giving consistency to other work products like the process diagram. Such a representation contains graphical and linguistic structures (eventual relationships, see Figure **3-9**).

A consistency mechanism is defined for representing events in the UNC-Method work products, which is based on event structures proposed in UNC-Method (Noreña, 2014). Event interaction graph (EIG) is used for representing event sequence by using PCS notation (Zapata *et al.*, 2013, 2014). Programming design patterns are defined in PCS to scientific software (Calle, 2016), which include mathematical functions defined by analysts (see Figure

3-21).

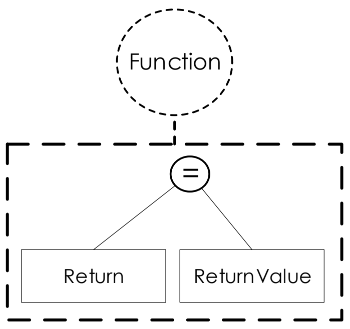


Figure 3-21 Mathematical functions in PCS defined by analysts (Calle, 2016)

We synthesize the PCS approaches for event represention and mathematical notation (5) in Table **3-4** according to the research questions (see Table **3-1**) of the literature review.

Table 3-5 PCS approaches for event represention and mathematical notation. The Authors

Authors	Approaches in PCS	Primary Studies	Modeling			Usability			Events				Mathematical notation				Time from events
			Type	Structural view	Behavioral view	Domain knowledge	Software development	simulation	Graphical structures	Linguistic structures	Mathematical structures	Functionality	Logical	Relational	Basic	Complex	
Chaverra, 2011	*Generación automática de prototipos funcionales a partir de PCS*	X	M	X	X	X	X						X	X	X		
Zapata, 2012	UNC-Method	X	Me	X	X	X	X		X	X	X		X	X	X		
Noreña, 2014	*Un mecanismo de consistencia para representar eventos disparadores y de resultado en UNC-Method*	X	Me	X	X	X	X		X	X	X		X	X			
Zapata *et al.*, 2013	Event Interaction Graph	X	D		X	X	X		X	X			X	X			
Zapata *et al.*, 2014																	
Calle, 2016	Programming design pattern in PCS for SSD	X	M	X	X		X						X	X	X	X	

The answer to *RQ1* is the following: commonly, events are represented in PCS approaches (3 out of 5) by using graphical, linguistic and mathematical structures in a multiple view of the system.

The answer to *RQ3* and *RQ4* is the following: All reviewed PCS approaches include logical, relational, and basic operators, and an approach includes complex operators for patterns; however, it lacks complex mathematical notation for representing events. Basic mathematical operations are involved by automated generation of prototypes (Chaverra, 2011) and UNC-Method (Zapata, 2012; 2 out of 5); however, such operations are only used in dynamic relationships.

The answer to *RQ5* and *RQ7* is the following: PCS lack time events and event functionality.

The answer to *RQ6* is the following: some approaches (4 out of 5) use the software components of the PCS for representing the domain knowledge and the logic system in software development; however, such schemas lack scientific components to be used in a simulation.

We summarize approaches from the reviewed four categories, whose notation includes linguistic structures for representing events in Table **3-6** for answering the *RQ2* (see Table **3-1**) of the literature review.

Table 3-6 Synthesis to linguistic structures. The Authors

Authors	Approaches	Linguistic structures			Consistency in verbs for events	Consistency in linguistic representation	List of verbs for events/ linguistic rules
		Eventual verb	Semantic role	Noun			
Haisjackl *et al.*, 2018 OMG, 2014a	BPMN	X	X				
OMG, 2015 OMG, 2011	UML Activity diagram	X	X				
Chonoles, 2017 OMG, 2015	UML State Machine diagram	X		X			X
OMG, 2015	UML Sequence diagram	X	X				
Amjad *et al.*, 2017 Xia *et al.*, 2014 Xue *et al.*, 2013	EPC	X	X		X		
Zapata *et al.*, 2014	EIG	X	X		X	X	X
Shekarpour *et al.*, 2019	CEVO	X	X	X	N/A	X	X
Etzion *et al.*, 2016	TEM			X	N/A	X	
Bauoya *et al.*, 2015	SysML State Machine			X	N/A	X	
Boubeta-Puig *et al.*, 2015; 2019	Medit4CEP			X	N/A	X	
Noreña, 2014 Zapata, 2012	PCS	X	X		X	X	X
He, 2016	Ontology-based vaccine and drug adverse event			X	N/A	X	
Liu *et al.*, 2016	Ontology pattern for emergency event modeling			X	N/A	X	
Patri *et al.*, 2014	The process oriented event model (PoeM) ontology			X	N/A	X	
Zhang *et al.*, 2015	Event ontology based on simple knowledge organization system (SKOS)		X	X	N/A	X	
Zhu, 2018	Quadruple anonymity trajectory (QAT) ontology			X	N/A	X	
Bazydlo *et al.*, 2014	UML state machine/finite state machine	X		A letter	X	X	
Armas-Cervantes *et al.*, 2016; Reinartz *et al.*, 2015	BPMN/Petri nets			A letter	N/A		
Ravikumar *et al.*, 2016	BPMN/ontology	X	X				
Bazhenova *et al.*, 2019	BPMN/decisión requirement diagram (DRD)			X	N/A	X	
Xia *et al.*, 2019	BPMN/modeling and analysis real-time, and embedded systems (MARTE)	X	X			X	
Dávid *et al.*, 2018	UML class diagram/finite automata			X	N/A		
Gao *et al.*, 2014	Ontology/syntax tree	X	X			X	

The answer to *RQ2* is the following: some approaches contains eventual verbs and semantic roles for linguistically representing events, *e.g.*, hold book in BPMN (Haisjackl *et al.*, 2018), new repair task arrived in EPC (Xue *et al.*, 2013), send entitlements, publish in website, prepare schedule, revise schedule, duration for revise, send revisions, etc. in BPMN/ontology (Ravikumar *et al.*, 2016). However, the verbs and representation are inconsistent, since sometimes the events are represented as objects (*e.g.*, EPC, UML, BPMN) while other events include verbs in the representation. Also, the events can be confused with processes/activities because they are represented with action verbs.

Also, some approaches include nouns for representing an event: "long call at night" (with a time preposition) in TEM (Etzion *et al.*, 2016); "body internal motion," "measure," "weather," etc. in CEVO (Shekarpour *et al.*, 2019); diseases as "thyroiditis," "influenza vaccine," etc. in the ontology-based vaccine and drug adverse event (He, 2016); "death of residents," "nausea," "cough," etc. in the ontology pattern for emergency event modeling (Liu *et al.*, 2016); and "pump" and "well failure" in PoEM ontology (Patri *et al.*, 2014). However, such nouns lack a verb/semantic role for completing what happen with such an event and who/what is affected by the event.

In addition, a list of verbs is proposed in CEVO for representing events *message* (ask, explain, teach, write, etc; see Figure **3-6**), but such verbs are action verbs, which are should have a semantic role *agent*, *e.g.*, the boss writes a message. List of events of PCS (arrive, emerge, arise, etc; see Figure **3-9**) is used with a semantic role *experiencer* (linguistic rule), *e.g.*, message emerges, such an event allows for knowing what happen by using a phrase, which contains both a noun and a verb. EIG is based on PCS notation (Zapata *et al.*, 2014). UML state machine diagram includes a linguistic rule with three elements *trigger event [guard condition] action* for representing an event, but it is inconsistently used, since only one/two elements are used, *e.g., [no reserve], [reserveOpen], returned* (action), *ReservePickup* (trigger event), *BorrowRequest[isCircBook]* (see Figure **3-3**). Also, the linguistic form for naming an event is by using the same action verb of a state *e.g., BorrowRequest* (trigger event) and *Borrowed book* (state, see Figure **3-3**).

3.3. Problem Statement

3.3.1. General Problem

Computer scientists have challenges for abstracting the problem as much as possible when their solutions should be used in several domains (Howison *et al.*, 2015). A gap between science and software engineering is cross-cutting to the scientific software development process in scientific software domains (Johanson & Hasselbring, 2018; Kanewala & Bieman, 2014; Wiese *et al.*, 2019; Wilson *et al.*, 2014).

According to the literature review, such a gap persists between event-driven software modeling and event-driven scientific modeling, due to the complexity of scientific software and the required specialized domain knowledge. Scientists often continue developing their scientific software (Kanewala & Bieman, 2014; Wilson *et al.*, 2014; Wiese *et al.*, 2019) for analyzing and simulating phenomena/events in a system. Such events are modeled by using scientific modeling approaches, which are based on mathematical models (commonly, such models are discretized) and terminology, but they lack software components—concepts, processes, events, and structures—of the domain knowledge in the same model, since scientists apply informal and non-standard software engineering practices in the implementation phase. Their science and engineering projects lack software documentation and requirements analysis processes and they reuse few models and code pieces (Johanson & Hasselbring, 2018; Wiese *et al.*, 2019). Scientific concepts (terminology) are inconsistent in the domain, since several names of variables, abbreviations, and quantities without units increase the difficulty for understanding the domain.

Business analysts use software modeling approaches in the analysis phase by applying standard software engineering practices. Such approaches include software components, which allow for representing events and other components of a domain. However, they lack scientific components—graphical, linguistic, and graphical complex structures—for representing mathematical equations, events, and event functionality (internal logic). Analysts also lack specialized domain knowledge. Such needs are unattended from the software engineering perspective (Johanson & Hasselbring, 2018).

Analysts have attempted to integrate both components in several scientific and software modeling approaches for analyzing phenomena/events in SSD. However, such integration is performed by using two and more models, which present inconsistency in domain components as concepts, quantities, and variables. Therefore, both *scientists and business analysts lack a computing model* with integrated scientific and software components for representing events in SSD from the analysis phase. We define the causes (C) of the general problem in a fishbone diagram (see Figure **3-22**).

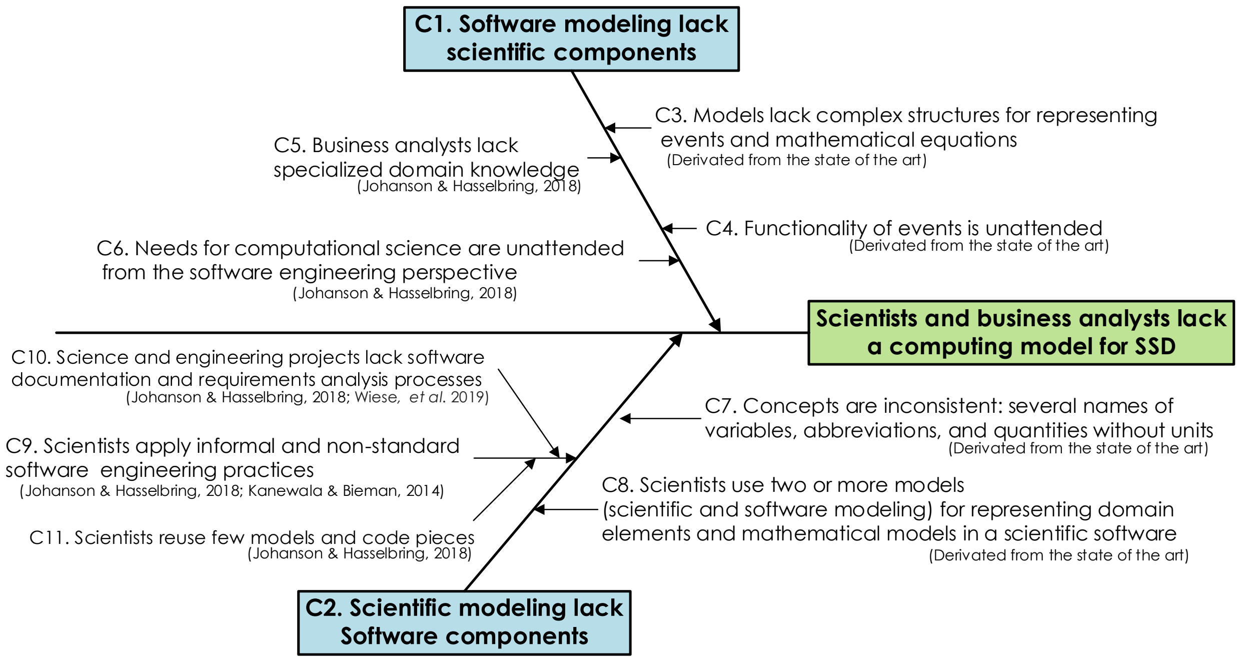

C1. Software modeling lack scientific components
C3. Models lack complex structures for representing events and mathematical equations
(Derivated from the state of the art)
C5. Business analysts lack specialized domain knowledge
(Johanson & Hasselbring, 2018)
C4. Functionality of events is unattended
(Derivated from the state of the art)
C6. Needs for computational science are unattended from the software engineering perspective
(Johanson & Hasselbring, 2018)
Scientists and business analysts lack a computing model for SSD
C10. Science and engineering projects lack software documentation and requirements analysis processes
(Johanson & Hasselbring, 2018; Wiese, et al. 2019)
C7. Concepts are inconsistent: several names of variables, abbreviations, and quantities without units
(Derivated from the state of the art)
C9. Scientists apply informal and non-standard software engineering practices
(Johanson & Hasselbring, 2018; Kanewala & Bieman, 2014)
C8. Scientists use two or more models (scientific and software modeling) for representing domain elements and mathematical models in a scientific software
(Derivated from the state of the art)
C11. Scientists reuse few models and code pieces
(Johanson & Hasselbring, 2018)
C2. Scientific modeling lack Software components

Figure 3-22 General problem. The Authors

3.3.2. Specific Problems

Pre-conceptual schemas lack scientific components for representing events in scientific software domains. Such a problem and its causes—event functionality, graphical, linguistic, and mathematical structures are incomplete—are common among the software modeling approaches. Nonetheless, PCS notation has advances in the event representation and the mathematical notation in the same model on opposite to other approaches in software modeling. A multiple (structural and behavioral) view of the system, notation for basic mathematical equations, mathematical (logic and relational), graphical, and linguistic structures (list of verb and semantic roles for events with linguistic rules and consistency in representation). However, PCS lack graphical, linguistic, and mathematical structures for representing events and their functionality in SSD. Such a representation is required for the analysis, development, and simulation process of scientific software. We define a set of causes of the specific problem according to PCS notation and the PCS approaches in a fishbone diagram (see Figure **3-23**).

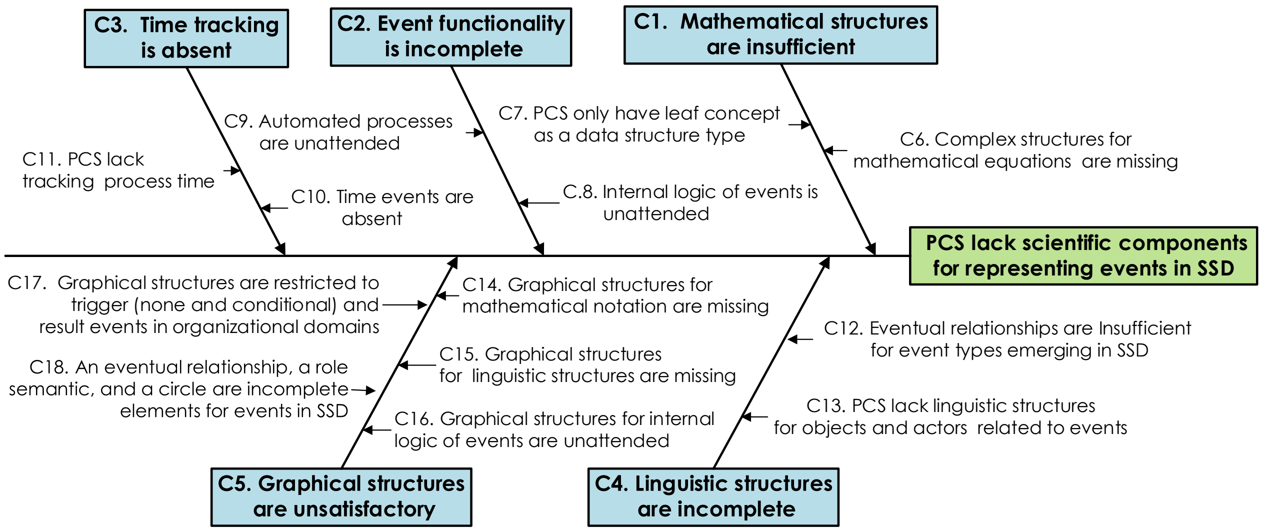

C3. Time tracking is absent
C2. Event functionality is incomplete
C1. Mathematical structures are insufficient
C9. Automated processes are unattended
C7. PCS only have leaf concept as a data structure type
C6. Complex structures for mathematical equations are missing
C11. PCS lack tracking process time
C10. Time events are absent
C.8. Internal logic of events is unattended
PCS lack scientific components for representing events in SSD
C17. Graphical structures are restricted to trigger (none and conditional) and result events in organizational domains
C14. Graphical structures for mathematical notation are missing
C12. Eventual relationships are Insufficient for event types emerging in SSD
C18. An eventual relationship, a role semantic, and a circle are incomplete elements for events in SSD
C15. Graphical structures for linguistic structures are missing
C13. PCS lack linguistic structures for objects and actors related to events
C16. Graphical structures for internal logic of events are unattended
C5. Graphical structures are unsatisfactory
C4. Linguistic structures are incomplete

Figure 3-23 Specific problems. The Authors

3.4. Research Question

How can we refine event representation and mathematical notation in scientific software domains by using pre-conceptual schemas?

RQ1. What structures are used for representing events in scientific and software modeling (view)?

RQ2. What linguistic errors are detected in events from models?

RQ3. What models include mathematical notation for representing events in scientific software domains?

RQ4. What mathematical structures are used?

RQ5. What models include the time by using events?

RQ6. What is used the model for?

RQ7. What models include event functionality?

3.5. Hypothesis

An extension to pre-conceptual schemas by using graphical, linguistic, and mathematical structures can be used for refining event representation and mathematical notation in SSD.

3.6. Objectives

3.6.1. General Objective

Refining event representation and mathematical notation by using an extension to pre-conceptual schemas.

3.6.2. Specific Objectives

- Characterizing events emerging from scientific software domains.
- Defining graphical, linguistic, and mathematical structures for event representation in PCS.
- Proposing an extension to PCS for the sake of event representation and mathematical notation refinement by using graphical, linguistic, and mathematical structures for representing events in scientific software domains.
- Validating the extension to PCS in an experiment in order to analyze the proposed structures understanding level.

We graphically summarize the objectives of this Ph.D. Thesis in Figure **3-24**.

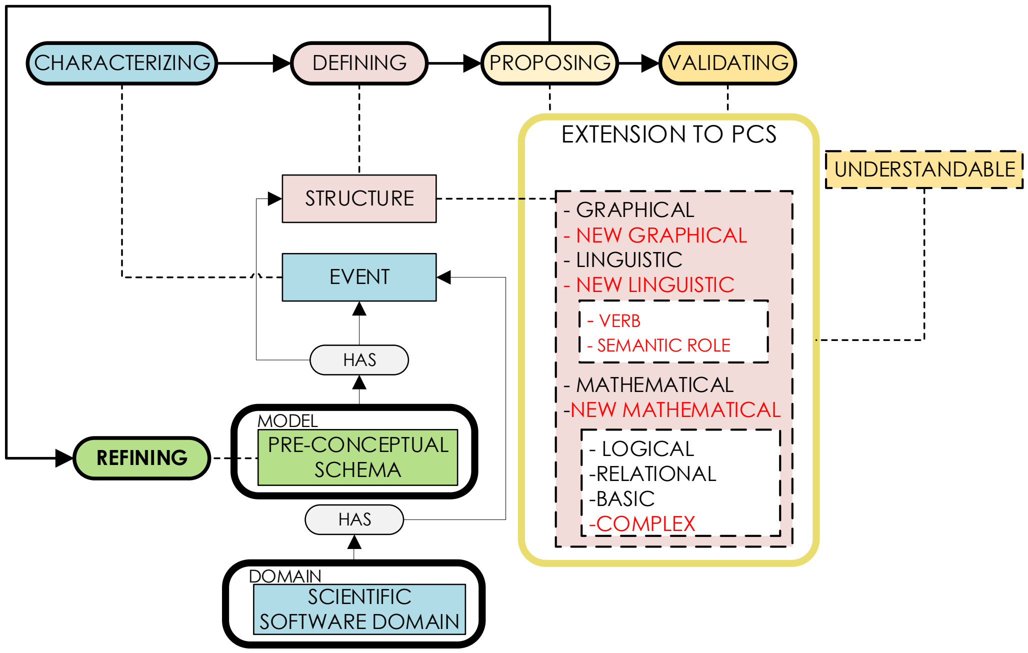


Figure 3-24 Objectives. The Authors

3.7. Justification

According to the *design science methodology for information systems and software engineering*, a *work product* is produced by designing an improvement to a problem; the *social context* contains the possible users of the work product; *the knowledge context* consists of existing theories from science and engineering, specifications of currently known designs, useful facts about currently available products, and lessons learned from the experience of researchers (Wieringa, 2014). In this Ph.D. Thesis, the work product performed is an *extension to PCS*; *businesss analysts, scientists*, and *students* integrate the social context, who can use the extended PCS as computing models in *scientific software domains* (knowledge context). Some key reasons for justifying the importance of this Ph.D. Thesis are:

Both software engineering and science fields are integrated in this Ph.D. Thesis. Extended PCS allow for integrating scientific and software components and reducing the gap between both fields.

PCS extension allows for refining event representation and mathematical notation in SSD.

PCS extension allows for representing time, event functionality, and structural and dynamic view of the elements of any SSD; understanding and recognizing of the processes, events, and mathematical models in a SSD.

Business analysts, scientists, and students can use the extended PCS as computing models for representing SSD and its elements in development and simulation processes.

We graphically summarize the justification in Figure **3-25**.

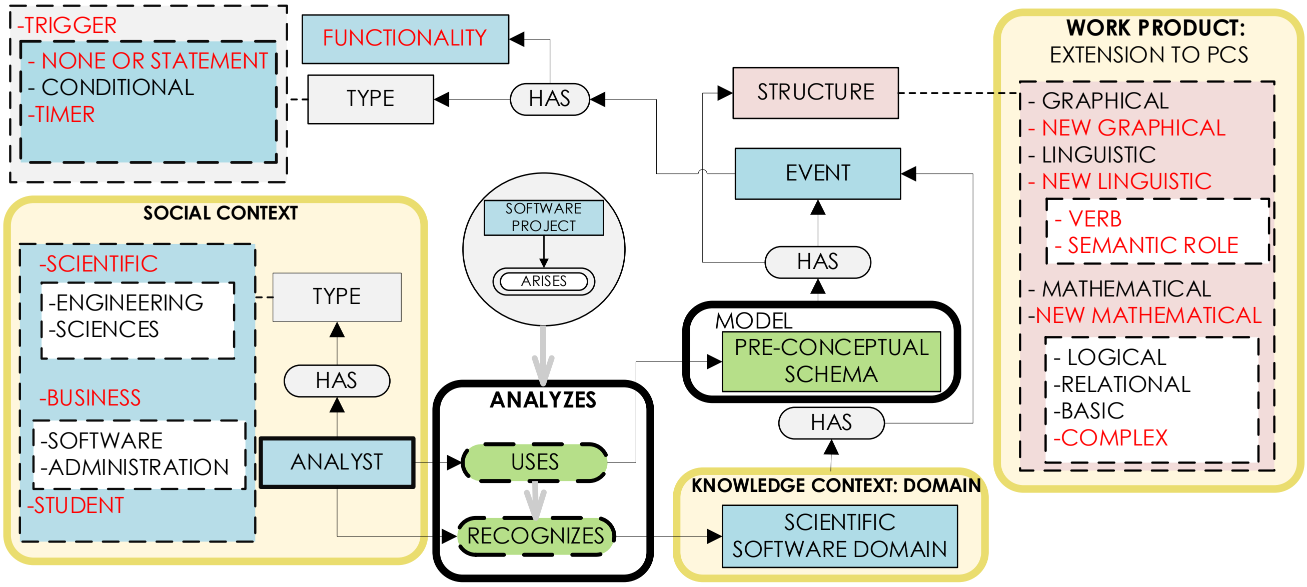


Figure 3-25 Justification. The Authors

4 Extension to PCS

Passion is an event that happens when both discipline and love are mixed for achieving better results.
—Paola Noreña

Pre-conceptual schemas include software components for representing a domain: processes, concepts, events, and structures—graphical, linguistic, and mathematical—which are domain knowledge elements used for understanding the system logic, analyzing requirements analysis, documentating the system, and code it. Computational linguistics rules are included in the PCS notation, which are focused on a relationship representation form, *e.g.*, a *concept*, and an *eventual relationship* for events. Analysts and stakeholders can easily understand the main components of the domain knowledge and developers can consistently code the software system (Zapata, 2012; Noreña & Zapata, 2018b).

Such components also allow for representing notation from scientific modeling: graphical, linguistic (eventual verbs and semantic roles for events with linguistic rules and consistency in the representation), and mathematical structures are used for representing events (Noreña, 2014); a multiple (structural and behavioral) view of the system; and a notation for basic mathematical equations as mathematical and graphical operators (logic, relational, and basic) and concepts.

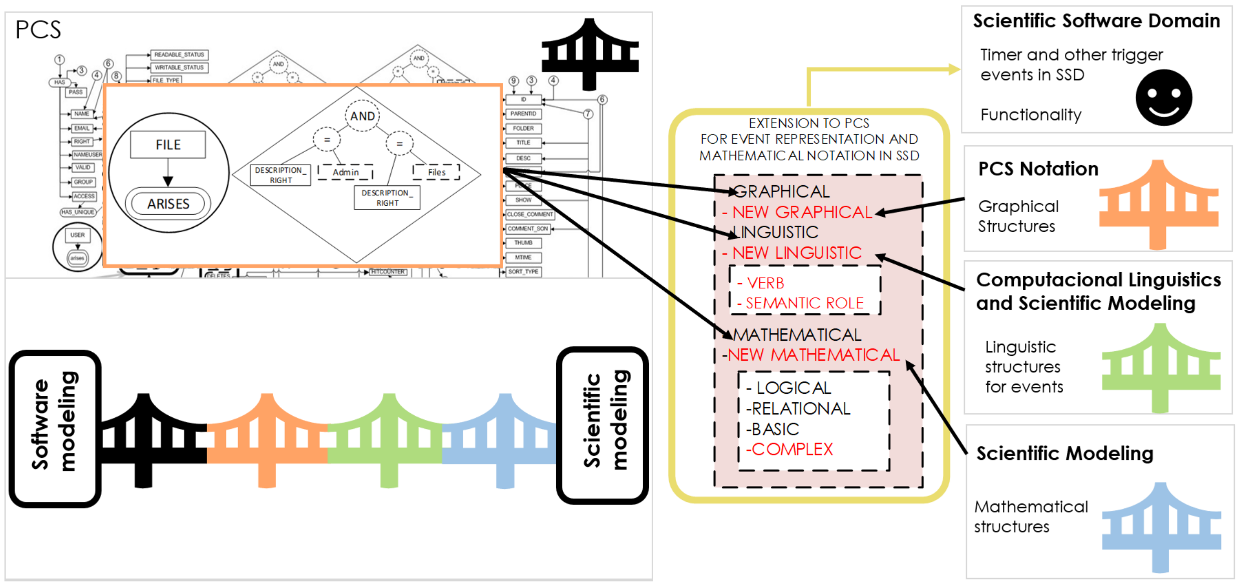
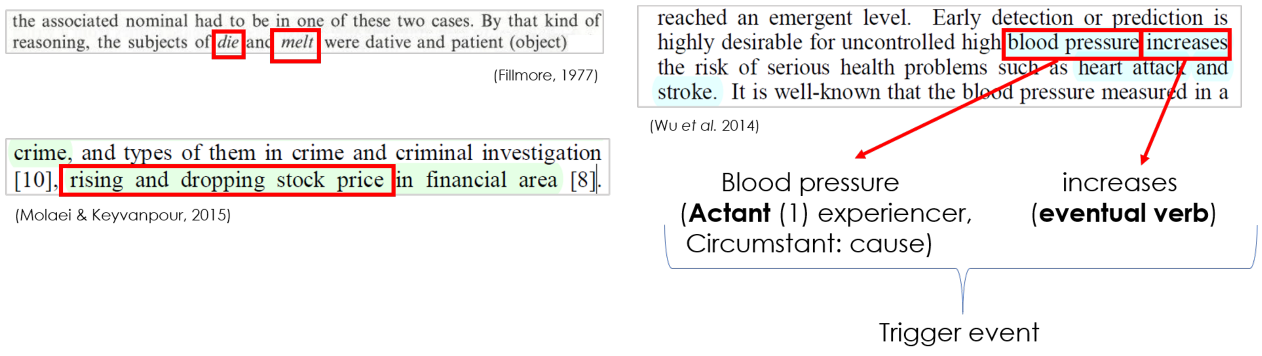


Figure 4-1 Proposal Solution. The Authors

An extension to PCS is proposed in this Ph.D. Thesis (see Figure **4-1**, see complete PCS in Figure **3-9**) for the sake of event representation and mathematical notation refinement in scientific software domains. Such refinement is performed for integrating scientific components: new linguistic structures from computational linguistics and scientific modeling, new mathematical structures from scientific modeling, and graphical structures from the PCS notation. Therefore, pre-conceptual schemas can be used as computing models with software and scientific components integrated into the same model for representing events (timer and other trigger events) and their functionality (internal logic) in SSD (see Figure **4-1**). We propose such an extension according to the research methodology in four steps: (i) we characterize events emerging from SSD, (ii) we define linguistic and graphical structures, (iii) we define mathematical and graphical structures for representing events in PCS, and (iv) we represent events in a SSD as lab study. Finally, we relate events represented in several SSD by using the extended PCS.

4.1. Characterizing Events emerging from SSD

We characterize events by using the following criteria: event in SSD and eventual verbs selected by linguists or philologists/eventual verbs used in SSD with semantic role different to an *agent*. Then, We identify expressions/phrases indicating an event in scientific and linguistic papers/books. We classify the eventual verbs by using semantic roles related to events—between zero to two actants, actants type *experiencer* (which experiment an event) and *patient* (who suffers the event effect) according to Fillmore (1977) and Gruber (1965); circumstant type *strengh* (whose origin is unknown and is produced by an event), and *cause* (situation generated by an event) according to Tesnière (1965) and Gruber (1965); such rules are defined from computational linguistics. Some examples for identifying and classifying eventual verbs are presented in Figure **4-2**. Event characterization is performed according to such a classification.


the associated nominal had to be in one of these two cases. By that kind of reasoning, the subjects of *die* and *melt* were dative and patient (object)

(Fillmore, 1977)

crime, and types of them in crime and criminal investigation [10], rising and dropping stock price in financial area [8].

(Molaei & Keyvanpour, 2015)

reached an emergent level. Early detection or prediction is highly desirable for uncontrolled high blood pressure increases the risk of serious health problems such as heart attack and stroke. It is well-known that the blood pressure measured in a

(Wu *et al.* 2014)

Figure 4-2 Event characterization in SSD. The Authors

4.2. Defining Linguistic and Graphical Structures for Event Representation in PCS

We define linguistic structures for extending event representation in a list of 38 eventual relationships. Such a definition and characterization with their semantic roles, and an example of events are presented into three categories: events with zero actants, events with one actant, and events with two actants. We define graphical structures for representing events by using the PCS notation and the found linguistic structures for each category.

4.2.1. Events with Zero Actants

Commonly, eventual verbs with zero actants are used for indicating natural events in scientific software domains as meteorology and climatology. We propose eventual relationships for such events in Table **4-1** (Noreña *et al.*, 2018), which do not require actants because the same verb has a complete meaning for expressing what happens *e.g.*, "rains" is the eventual relationship. Natural events are caused by weather changes or cycles and they can generate other events. Circumstant type of such events is a *cause*.

Table 4-1 Linguistic structures for events with zero actants (Noreña *et al.*, 2018)

Eventual Relationship	Author	Example of event	Scientific software domain	Semantic Role		
				Actant		Circumstant
				Quantity	Type	
1. Rain	Tesnière, 1965	It rains	Meteorological, Climatological	0	Not required	Cause
2. Thunder	Dayeh *et al.*, 2015	It thunders				
3. Hail	Burcea *et al.*, 2016	It hails				
4. Snow	Zapata, 2012	It snows				

We define a graphical representation by using an *eventual relationship* (without a concept) in a circle consistently with the linguistic structures for this event category. We show some examples in Figure **4-3**.

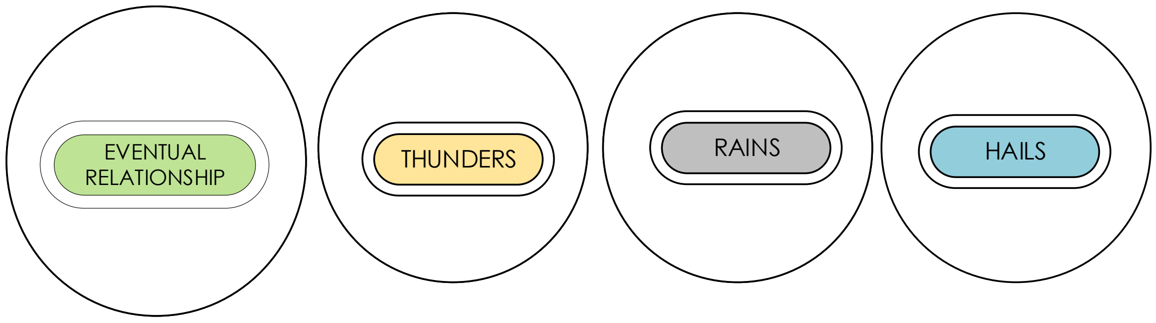


Figure 4-3 Graphical structures for events with zero actants (Noreña *et al.*, 2018)

4.2.2. Events with One Actant

Most events are linguistically represented by using an eventual verb accompanied by an actant for expressing their meaning in SSD. Circumstants of events with one actant are *strength* and *cause* since they can be generated by other events; however, the origin is unknown and also cause, other events (Gruber, 1965). We identify several events in this category: discrete events (*e.g.*, sensor alarm sounds, time passes), deterministic events (*e.g.*, measure appears), and non-deterministic events (*e.g.*, new data arrives, customer arrives).

Nouns are also used for representing natural events (*e.g.*, earthquake, environmental noise) and diseases (*e.g.*, cancer), or symptoms (*e.g.*, vomit, nausea, headache). When an event is identified as a noun, *e.g.* pollution, such an event needs a verb for completing its meaning *i.e.*, pollution increases; the noun "pollution" is used as an actant with the eventual relationship "increases" for indicating what happened.

We propose eventual relationships for events with one actant in Table **4-2**, *e.g.*, voltage rises, "voltage" is an actant type *experiencer* and "rises" is the eventual relationship (Noreña *et al.*, 2018); such an event has circumstant types *strength* and *cause* because a strength could generate and cause other events (*e.g.*, electric current increases).

We define a graphical representation with three concept types for events with one actant according to the actant types *experiencer* and *patient*: a *concept* (*e.g.*, volcano erupts), a *class concept* (*e.g.*, sensor alarm sounds, "sensor" is a class and "alarm" is its leaf concept (attribute) and a *variable* (*e.g.*, time passes, "time" is an independent variable) for being used with an *eventual relationship* as we show in Figure **4-4**.

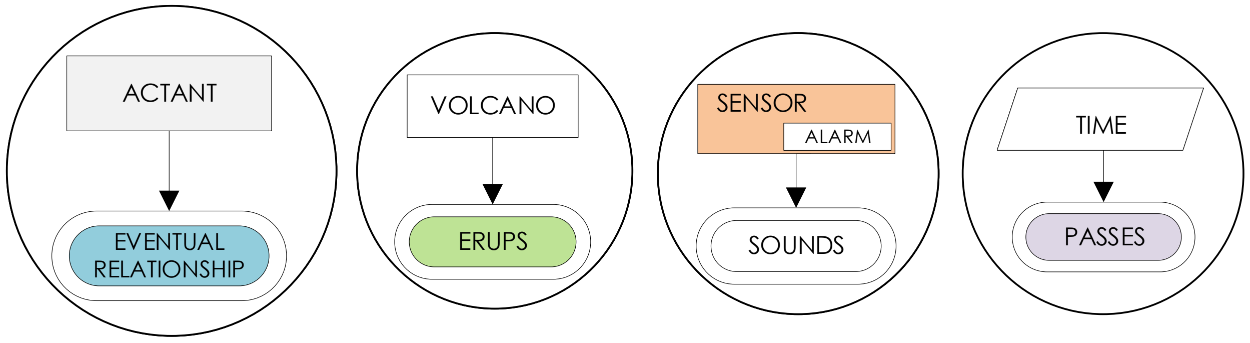


Figure 4-4 Graphical structure for events with one actant (Noreña *et al.*, 2018)

Table 4-2 Linguistic structures for events with one actant (Noreña *et al.*, 2018)

Eventual Relationship	Author	Example of event	Scientific software domian	Semantic Role		
				Actant		Circumstant
				Quantity	Type	
5. Rise	Molaei & Keyvanpour, 2015	Stock price rises; Voltage rises	Industrial Electronic			
6. Increase	Wu *et al.*, 2014; Beltrán, 2015	Blood pressure increases; Dollar price increases	Medical Economic		Experiencer	Strength, cause
7. Grow	Merkens *et al.* 2016	Population grows	Statistical			
8. Decrease	Beltrán, 2015	Dollar price decreases	Economic			
9. Drop	Garrudo, 1990; Molaei & Keyvanpour, 2015	Sale drops	Industrial	1		
10. Sneeze	Zapata, 2012	Patient sneezes	Medical			
11. Bleed	Paddock & Chapin, 2016	Patient bleeds	Medical			
12. Convulse	Taiwe *et al.*, 2016	Patient convulses	Medical		Patient	Cause
13. Die	Fillmore, 1977; Zapata, 2012	Animal dies	Biological			
14. Sleep	Gruber, 1965; Zapata, 2012	Patient sleeps	Medical			
15. Tinkle	Zapata, 2012	Cellphone tinkles	Electronic			
16. Sound	Dayeh *et al.*, 2015	Thunder sounds	Physical			
17. Ring	Tarun *et al.*, 2017	Alarm rings	Industrial			
18. Fly	Garrudo, 1990; Zapata, 2012	African bee flies	Biological			
19. Fall	Gruber, 1965; Tesnière, 1965	Lightning falls	Meteorological	1	Experiencer	Cause
20. Arrive	Dayeh *et al.*, 2015 Zapata, 2012	Wave arrives; Cholesterol arrives	Physical Chemical			
21. Emerge	Zapata, 2012	Bacteria emerges	Bacteriological			
22. Come	Zapata, 2012	Signal comes	Electronic			
23. Appear	Kuznetsov & Merzlikin, 2019	Electric wave appears				
24. Arise	Zapata, 2012	Customer arises	Industrial			
25. Erupt	White & McCausland, 2016	Volcano erupts	Vulcanological			Strength, cause
26. Melt	Fillmore, 1977	Lava melts				
27. Boil	Zapata, 2012	Water boils	Geological			
28. Expire	Baouya *et al.*, 2015	Product expires	Industrial			
29. Start	Herzberg *et al.*, 2013	Service starts				
30. Pass	Zapata, 2012	Time passes	Meteorological	1	Experiencer	Cause
31. Happen	Fillmore, 1971; Meng *et al.*, 2014	Hurricane happens	Geological			
32. Occur	Fillmore, 1971; Lukham, 2011	Earthquake occurs				
33. Change (increase and decrease)	Vose *et al.*, 2017	Temperature air conditioner changes	Electronic			

4.2.3. Events with Two Actants

We identify events linguistically represented for an eventual verb accompanied by two actants for expressing their meaning in SSD. Such events are predominant in the medical domain.

We propose eventual relationships for events with two actants in Table **4-3**, *e.g.*, patient suffers heart attack "patient" is an actant type *patient*, "suffers" is the eventual relationship, and "heart attack" is a second actant, which is used for completing the meaning about what happened to the patient. We also identify in this category, the eventual relationships *increases* and *decreases* (which also is in the category *events with one actant*), *e.g.*, temperature increases water pressure; "water pressure" is an actant type *experiencer*.

Table 4-3 Linguistic structures for events with two actants (Noreña *et al.*, 2018)

Eventual Relationship	Author	Example of event	Scientific software domain	Semantic Role		
				Actant		Circumstant
				Quantity	Type	
34. Suffer	Gruber, 1965	Patient suffers hemorrhage				
35. Present	Drăghici *et al.*, 2018	Patient presents abdominal pain	Medical		Patient	
36. Block	Zhao *et al.*, 2017	Lipid blocks vein				
Increase/ Decrease	Wu *et al.*, 2014; Beltrán, 2015	Temperature increases water pressure	Hydraulic	2	Experiencer	Cause
37. Loss	Obi *et al.*, 2018	Patient loses weight	Medical		Patient	
38. Gain	Obi *et al.*, 2018	Patient gains weight; Dollar gains price	Medical Economic		Patient Experiencer	

We propose a graphical representation of events with two actants and present some examples in Figure **4-5**. Some events are represented in this category, which include prepositions *in, on, at, to* in the eventual relationship, *e.g.*, epidemy arrives at city. In this case, the preposition should be used with the eventual relationship (see such an event in Figure **4-5**).

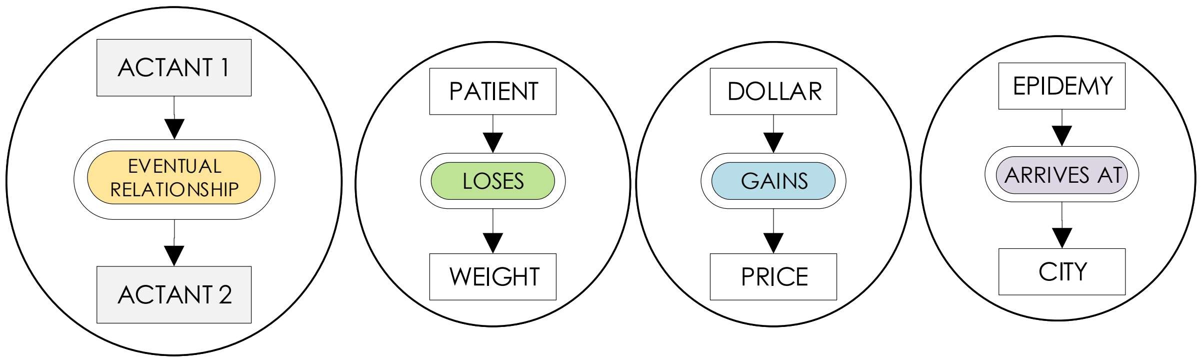


Figure 4-5 Graphical structures for events with two actants (Noreña *et al.*, 2018)

An event can be graphically and linguistically represented in equivalent forms according the perspective of an analyst business/scientist and related to linguistic rules for events in PCS. *e.g.*, the event *patient suffers hemorrhage* can be also represented as *patient bleeds* and *hemorrhage appears* (see Figure **4-6**).

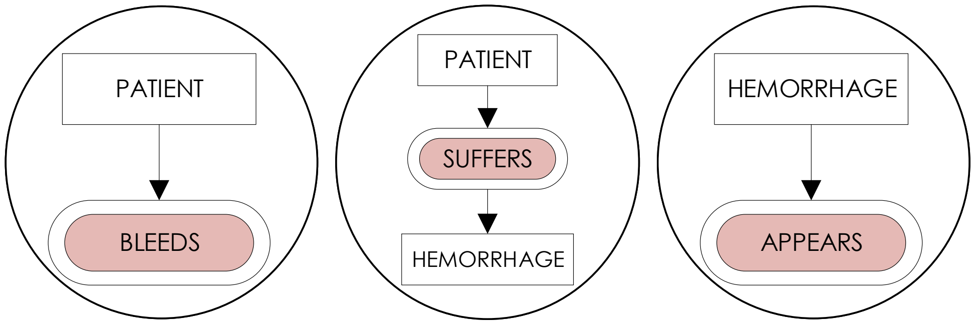
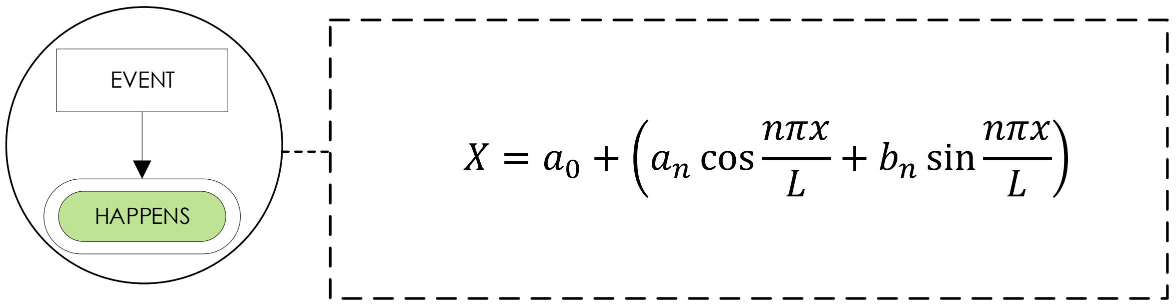


Figure 4-6 Equivalent forms of event representation (Noreña *et al.*, 2018)

4.3. Defining Mathematical and Graphical Structures for Event Representation in PCS

Event functionality contains the internal logic, which is formulated by using mathematical equations and conditions in a system in order to analyze a phenomenon and its behavior. We propose a representation of the event functionality by using a *specification* or a *constraint* (from PCS notation, see Figure **2-1**) linked to an event as we show in Figure **4-7**. Specification or constraint related to an event should contain such elements and the domain knowledge should be in the same model for a better understanding the context. We define the mathematical notation and event representation in four steps for integrating such scientific elements: (i) we characterize the elements of equations used in the internal logic of events identified in scientific modeling; (ii) we define mathematical and graphical structures by using the PCS notation for representing mathematical equations; (iii) we represent mathematical equations in PCS; and (iv) we represent events from SSD with proposed notation.



$$X = a_0 + \left(a_n \cos\frac{n\pi x}{L} + b_n \sin\frac{n\pi x}{L}\right)$$

Figure 4-7 Functionality of events in PCS. The Authors

4.3.1. Characterizing Elements of an Equation

Mathematical equations are *self-contained*, *i.e.*, the equations integrate elements of a context/domain in the operation. Element understanding should be obtained from either the context documentation or previous knowledge acquired by a scientist (Noreña & Zapata, 2018a).

Translating from Equation Symbols to Conceptual Form

We select as an example the *Malthus growth law* presented in the Equation 4-1, which is applied to the scientific domain *statistics* according to the text in Figure **4-8**.

$$y(t) = y0.\, e^{r(t-t0)} \tag{4-1}$$

We search the meaning of every element (symbol) of the equation in the context documentation, which is used for finding concepts related to the elements, *e.g.*, the element *y(t)* is *Population Value* according to the text in Figure **4-8**. We translate the original form of the equation (see Equation 4-1) to a conceptual form (see Equation 4-2). Commonly, initial conditions are also defined in the context documentation.

$$Population\, Value = Initial\ Population.\, e^{growth\, rate(time-start\ time)} \tag{4-2}$$

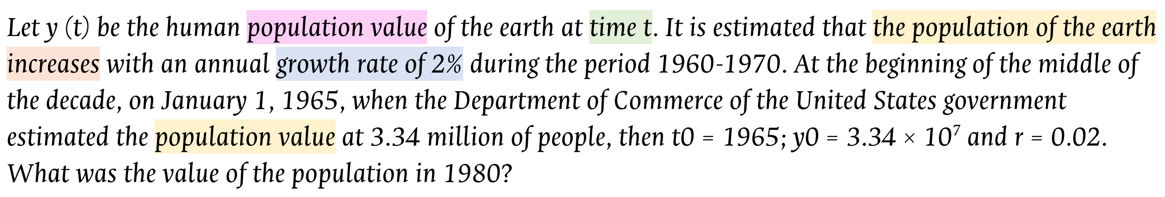
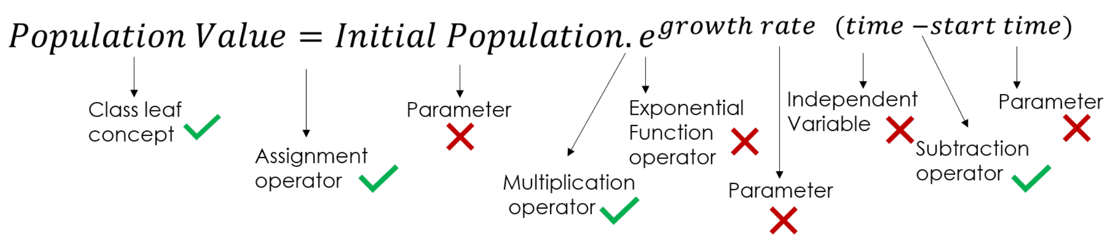

Let y (t) be the human population value of the earth at time t. It is estimated that the population of the earth increases with an annual growth rate of 2% during the period 1960-1970. At the beginning of the middle of the decade, on January 1, 1965, when the Department of Commerce of the United States government estimated the population value at 3.34 million of people, then t0 = 1965; y0 = 3.34×10^7 and r = 0.02. What was the value of the population in 1980?

Figure 4-8 Context documentation. The Authors translated from Navas (2017)

Identifying Elements of an Equation

We analyze the element type from the translated equation (see Equation 4-2) and compare them with elements of the PCS notation. Then, we identify what elements are required for integrating them in such a notation as we show in Figure **4-9**, we identify class (*population*), leaf concept (*value*), assignment, multiplication, and subtraction operator as elements present in the PCS notation while such a notation lacks parameters (*initial population, growth rate*, and *start time*), exponential function operator, and an independent variable. Other identified elements are initial conditions, arrays (vector, matrix, independent), mathematical, arrays, and trigonometric operators.



Population Value = Initial Population. e^growth rate (time − start time)
Class leaf concept ✓
Assignment operator ✓
Parameter ✗
Multiplication operator ✓
Exponential Function operator ✗
Parameter ✗
Independent Variable ✗
Subtraction operator ✓
Parameter ✗

Figure 4-9 Element Identification. The Authors

4.3.2. Defining Mathematical Notation in PCS

We define a set of mathematical structures identified in equations from SSD and propose graphical structures for representing such elements in PCS notation.

Nodes

We extend nodes from PCS notation based on the element *concept* (see Figure **2-1**) for representing the terminology used in scientific software domains.

Parameter is used for representing a constant value of an equation/function. We define a hexagon-shaped structure for representing a parameter in PCS notation (see Figure **4-10**). Such a parameter from PCS should have a constant value in any time of the system/simulation (Noreña & Zapata, 2018a; Calle *et al.*, in process). *e.g.*, *Pi* number equal to 3.1415 in Figure **4-10**.

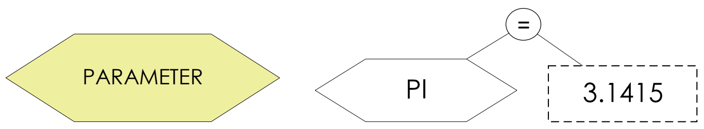


Figure 4-10 Parameter. The Authors

Independent variable is a non-dependent value of other variables and it is used for controlling dependent variables (which can be represented with the element *concept*) in a domain. We define a parallelogram-shaped structure for representing an independent variable (see Figure **4-11**). An independent variable from PCS should have a specific name and be used for controlling other variables in the system (Noreña & Zapata, 2018a; Calle *et al.*, in process). *e.g.*, a variable *valve* whose values are “closed” and “open” for controlling a liquid flow.

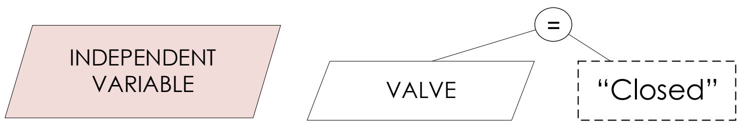


Figure 4-11 Independent variable. The Authors

Arrays are structures used for storing several values of a variable. Usually, arrays are used in the systems as a data structure for storing values in memory. We define two types of arrays:

Dependent array is a *vector* or *matrix* related to a class. We define an element *concept* from the PCS notation and add from one to two rectangles in its upper corner with a *term* value for representing a dependent array (see vector and matrix in Figure **4-12**). *e.g.*, class *measure* has a vector *value* and a matrix *block* in Figure **4-12**.

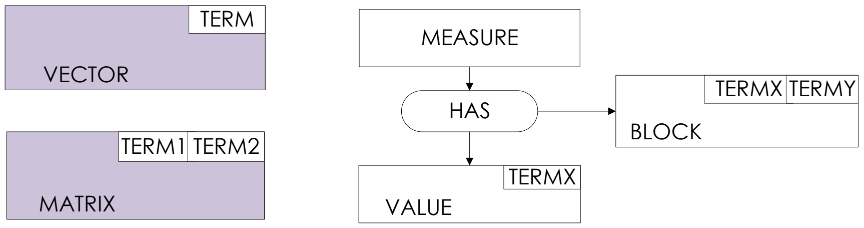


Figure 4-12 Dependent arrays. The Authors

Independent array is a non-dependent array of a class. We define the independent variable by using a parallelogram accompanied by one to two terms in its upper corner (see Figure **4-13**). *e.g.*, vector *space* in Figure **4-12**.

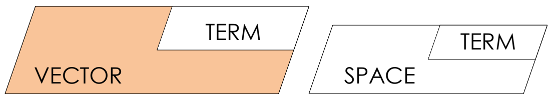


Figure 4-13 Independent arrays. The Authors

Term is used for defining the position of each element into the array and size step of the array. A vector should have one dimension and a matrix should have two dimensions, then a vector also requires one *term* and a matrix requires two *terms* respectively (Calle *et al.*, in process).

Gatherer

We extend the gatherers for completing start values used in software development and simulation process.

Initial Conditions are specifications including variables and parameters for beginning the simulation of a system. We define initial conditions by using the element *specification* from the PCS notation (see Figure **2-1**) accompanied by the name **initial conditions** (see Figure **4-14**). Such a specification should include variables, parameters, and functions of such variables and parameters (Noreña & Zapata, 2018a;

Calle *et al.*, in process). *e.g.*, the parameter *Pi* and variable *valve* inside of initial conditions in Figure **4-14**.

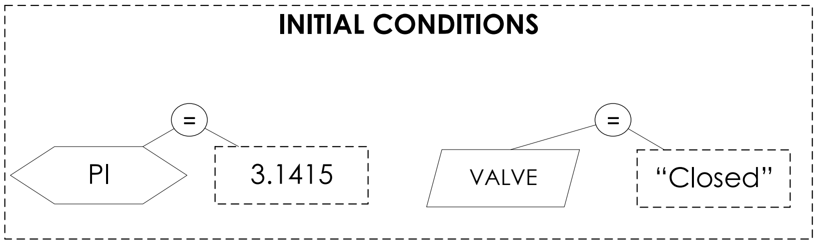


Figure 4-14 Initial conditions. The Authors

Operators

We extend the element *operator* of PCS notation (see Figure **2-1**) by including a set of new mathematical, trigonometrical, and array operators, which are predefined and commonly used in mathematical models. Operators are used with a value as argument, which should be a concept, a note-value, and a parameter (Calle *et al.*, in process).

Mathematical operators are elements used in complex equations (see Figure **4-15**). *Sqrt* operator is defined for representing the square root operation. *Exp* operator is defined for representing the exponential function. *Log* operator is proposed for representing the logarithm mathematical function. *Abs* operator is proposed for returning the absolute value of either a concept or a parameter (Calle *et al.*, in process).

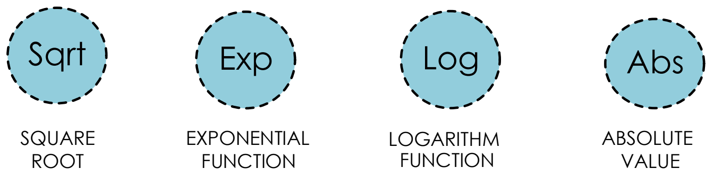


Figure 4-15 Mathematical Operators (Calle *et al.*, in process)

Array operators are used for inserting (*Push*) and for removing (*Pop*) values (see Figure **4-16**) into the last position of dependent and independent arrays (Calle *et al.*, in process).

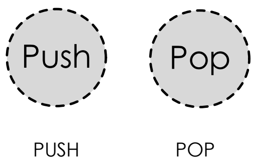


Figure 4-16 Array Operators (Calle *et al.*, in process)

Trigonometric operators are *Sin, Cos, Tan, Csc, Ctg*, and *Sec* operators (see Figure **4-17**), which are proposed for representing the trigonometric function sine, cosine, tangent, cosecant, cotangent, and secant respectively (Calle *et al.*, in process).

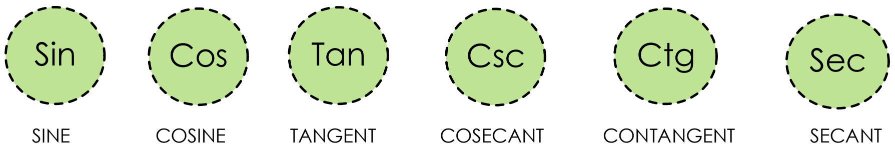


Figure 4-17 Trigonometric Operators (Calle *et al.*, in process)

We show an example in Figure **4-18** for observing how an operator should be represented, a *sin* operator is related to a concept (*amplitude*) and with other operator (*multiplication*).

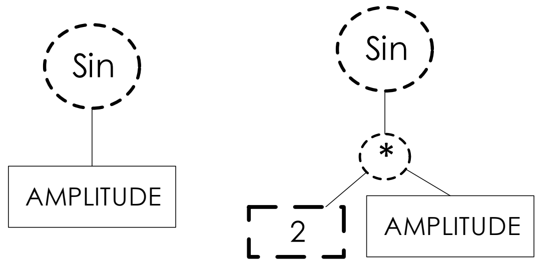


Figure 4-18 *Sin* operator. The Authors

4.3.3. Representing Equations in PCS

We follow the example of Equation 4-2 translated from Equation 4-1. We use the extended new mathematical structures for representing the equation in PCS notation (see Figure **4-19**, the color are used for explaining and guiding the traceability and consistency of the elements).

$$\mathit{Population\ Value} = \mathit{Initial\ Population}.\,e^{\mathit{growth\ rate}(\mathit{time}-\mathit{start\ time})}$$

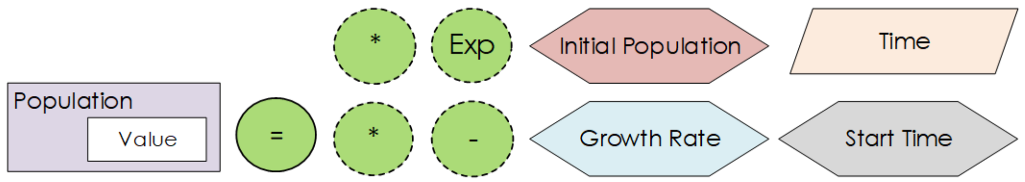


Figure 4-19 Equation symbols in PCS notation. The Authors

We use the binary expression tree, commonly used for representing algebraic and Boolean expressions, *e.g.*, a binary expression tree for the polynomial $2y + w^2z + wx + wy + wz$ in

Figure **4-20** (Kuipers *et al.*, 2015). Such a tree is also used in basic mathematical operations from PCS (Chaverra, 2011). Mathematical structures are related to the link *operator* (see Figure **2-1**) and the equation is completed in Figure **4-21**. Result values in an equation should have a unit for saving consistency.

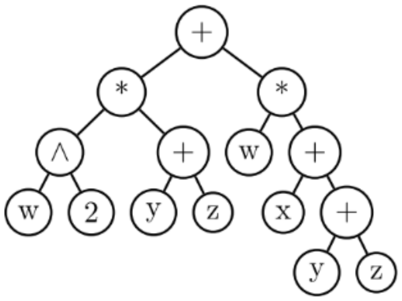


Figure 4-20 Binary expression tree (Kuipers *et al.*, 2015)

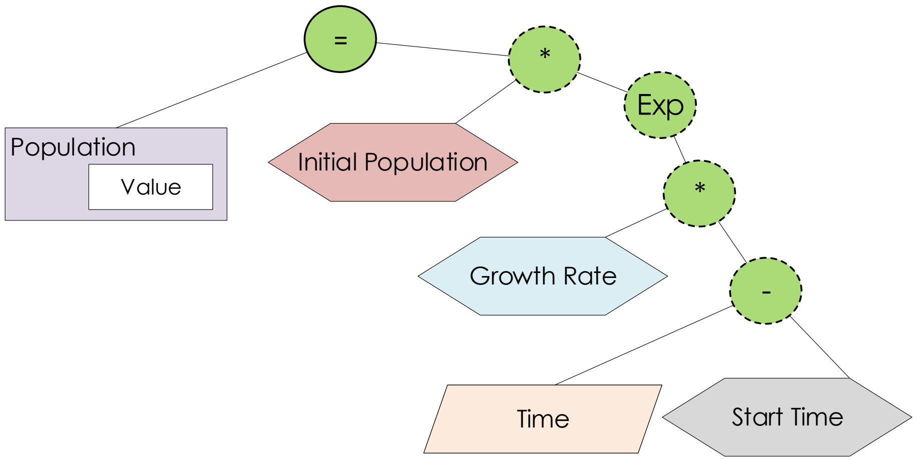


Figure 4-21 Equation in PCS notation. The Authors based on Calle *et al.* (in process)

4.3.4. Representing Events in PCS

Event Functionality

We extend the event representation in PCS by using the gatherers *specification* and *constraint* (see Figure **2-1**), which should be linked (link *concept-note*) to an *event* for representing the event functionality and analyzing its internal logic in SSD as we show in Figure **4-7**. Such gatherers should contain at least a dynamic operation—read, insert, update, delete—in either data bases or data structures—vectors and matrices. Operations are represented by using the *dynamic relationship* symbol without an agent (semantic role, since it should express an automated process/phenomenon) exclusively when it is inside the *specification* (without conditions) and *constraint* (with conditions) of an event. Equations are included by using another *specification/constraint* linked to the dynamic operation (according to the notation used in basic mathematical operations), *e.g.*, the event *population.value increases* is represented with a dynamic operation *inserts population.value* in Figure **4-22**—linked to the equation represented in Figure **4-21**—for analyzing the values each year according to the context documentation in Figure **4-8**.

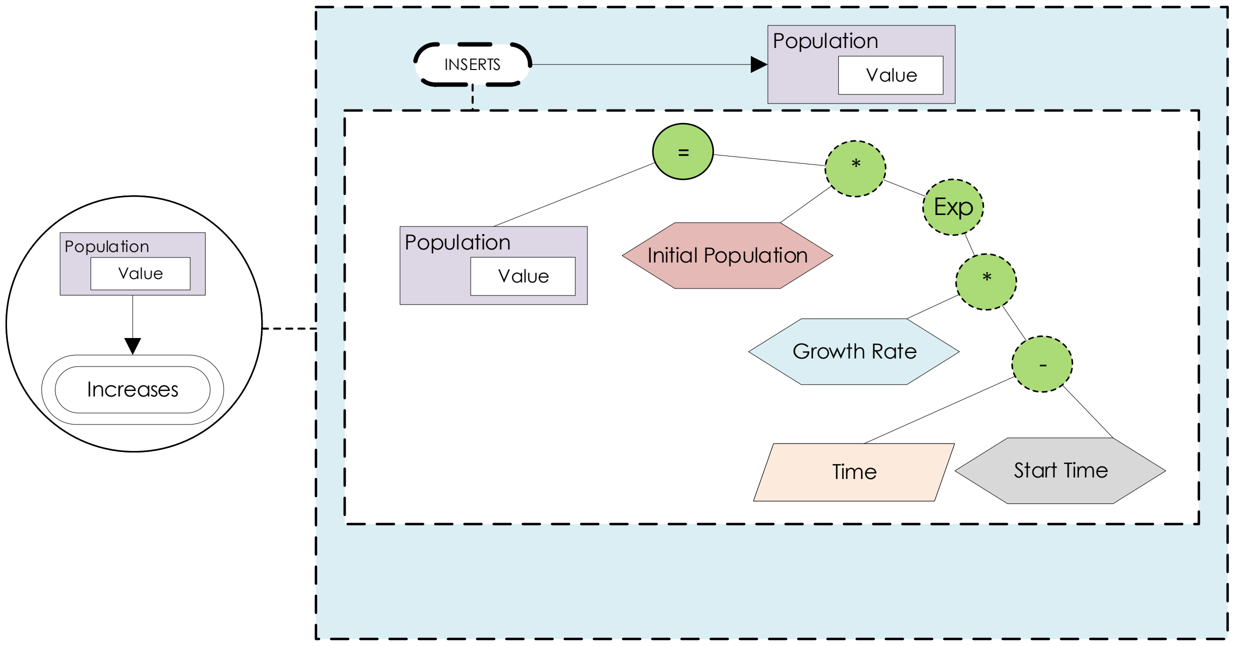


Figure 4-22 Event functionality in PCS notation. The Authors

Timer

Timer is a time event, which is required in the event functionality for simulating the system in a SSD and tracking its phenomena and processes. We represent a timer by using a cycle with the operator *sum* for increasing the time value (See Figure **4-23**), which can have conditions according to the domain. Commonly, parameters and variables are used in the internal logic of the timer, *e.g., time = 0 weeks* in order to know the start time and *end time = 244 weeks* for ending the time in Figure **4-23**. When a *specific date* is required, *e.g.*, *open time = 9:00*, *close time = 17:00*, they should be represented as parameters in *initial conditions* and they can be used in conditions of the system *e.g.*, if *close time = 17:00* then *door state = closed*, in the event *time passes* if *time >= open time and time <= close time.*

Time value can have digital format and time units—hours, minutes, seconds, weeks, years, etc. If the time expression is a specific value, *e.g.*, 1965 according to Figure **4-8**, it does not require units. Incremental value can be used according to the domain from *1* to *1* as we show for a value without units, *e.g.*, if *time = 1965, time = time + 1* and values with units, *e.g.*, if *time = 0 seconds, time= time + 1 seconds* in Figure **4-24**. Incremental value can be also used from other incremental values, *e.g.*, from *100* to *100*, if *time= 400 years, time = time + 100 years* as we show in Figure **4-23**.

We include a *timestamp* for controlling the arrival time of an event and obtaining the change states of the system (Luckham, 2002). We propose a *timestamp* as a variable, which has two states *"next"* and *"stop"* allowing to stop and continue the time of the system, *e.g., timestamp* used in *time passes* and *initial conditions* in Figure **4-23**.

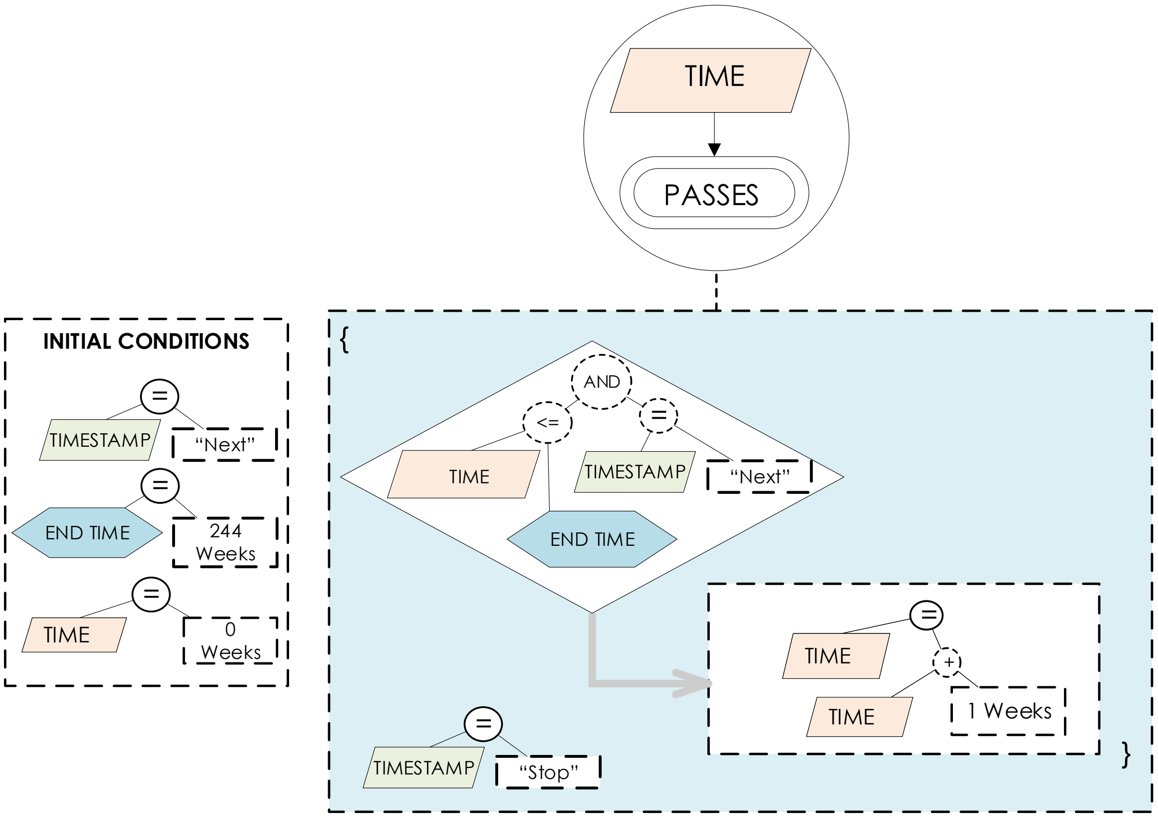


Figure 4-23 Timer. The Authors

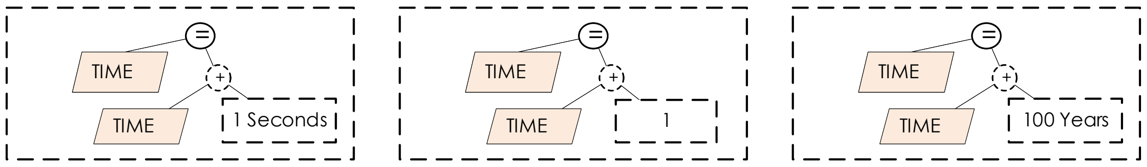


Figure 4-24 Timer values. The Authors

We integrate *initial conditions*, *timer*, and other *events* about phenomena for assuring the completeness of the event functionality in a simulation of the system. We follow the example of the statistics domain in Figure **4-8** and include its representation of Figure **4-22**; the *initial conditions* and *timer* are added in Figure **4-25**.

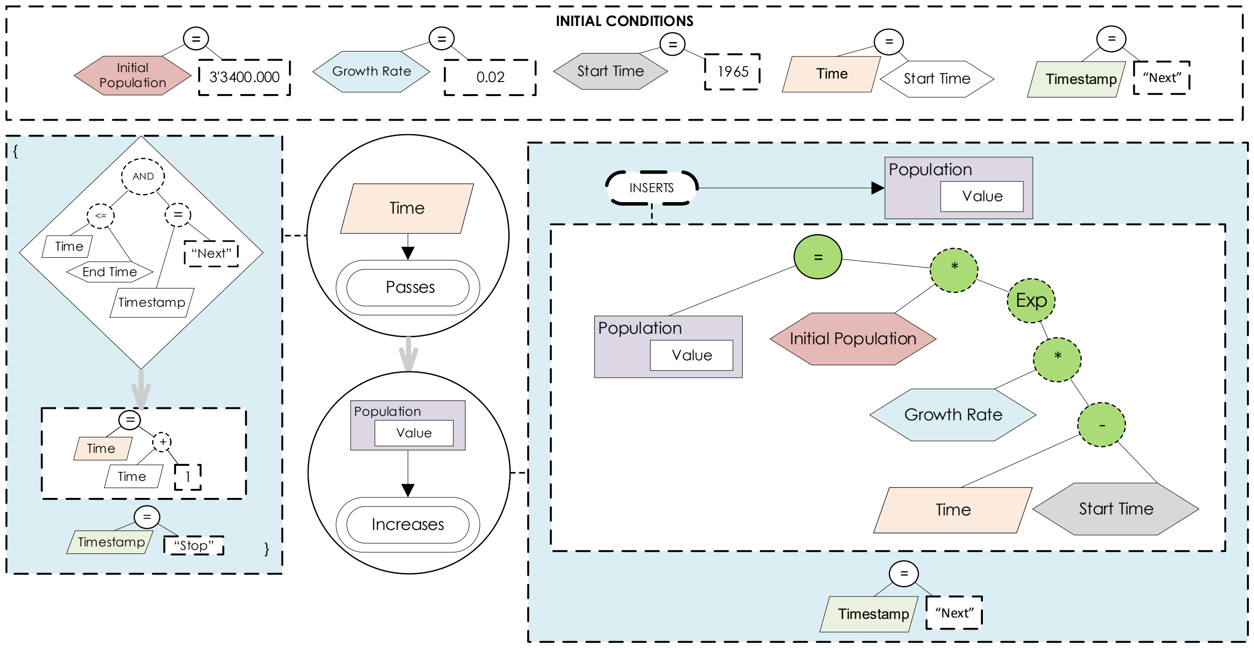


Figure 4-25 Event functionality with *initial conditions* and *timer*. The Authors based on Calle *et al.*, in process

Parameters *initial population, growth rate, start time*, and variable *time* are required by the functionality of the equation, which are represented in the *initial conditions*; timer *time passes* allows for increasing the *time* and triggers the event *population.value increases*, since *population.value* is dependent on the *time* for increasing such a value.

Variable *timestamp* is used for controlling the system, when the time changes, *timestamp* is equal to *"stop"* for inserting a new *population.value*. After the insertion, *timestamp* changes to *"next"* for going on the simulation.

4.4. Lab Study

We apply the extended PCS by using a lab study in the chemical SSD (see Figure **4-33**). Level of detail of the model is focused on the analysis of chemical events of mixture and concentration of substances in a software development and simulation process. We select the following context documentation, which is translated from Navas (2017):

> A *container of 300 liters* is full in two thirds of its *capacity* and contains *50 Kg* of salt. Valves are opened in *time t = 0 minutes*. A salt solution is added with a *concentration*

of one third of a kilo per liter to container with a *velocity of 3 liters per minutes.* If the *mixture* is extracted from the container with a *velocity of 2 liters per minutes.* How many kilograms of salt are found in the container?

Let *y (t)* be the *amount of salt* in the container in the minute *t.* The reason for changing in every minute *y'(t)*, it will be equal to the *amount of salt* entering to the container, minus the amount of salt coming out in the same minute. *Entry velocity of salt* is *1/3 Kg/Liters x 3 liters/minutes = 1Kg/minutes. Exit velocity* is calculated by the following: for the minute *t*, *y(t)* is *200 + t liters of water.* It is *2y(t)/(t + 200) kilograms of salt in* 2 liters. Consequently:

$$\begin{gathered} y'(t) = 1 - \frac{2}{t+200}y(t), \quad \text{it is a differential equation} \quad y'(t) + 1\frac{2}{t+200}y(t) = 1, \\ \text{its integrating factor is} \quad \mu(t) = e\int \frac{2}{t+200}dt = e^{2ln(t+200)} = (t+200)^2 \\ \text{integrating} \quad (t+200)^2 y = \frac{(t+200)^3}{3} + C \Rightarrow \quad y(t) = \frac{t+200}{3} + \frac{C}{(t+200)^2} \end{gathered} \tag{4-3}$$

The particular interest is the *initial condition y(0) = 50.*

$$\begin{gathered} \text{Replacing in the equation:} \quad 50 = \frac{200}{3} + \frac{C}{(200)^2} \Rightarrow \quad C = \frac{-50}{3}200^2 \\ \text{the solution required is} \quad y(t) = \frac{t+200}{3} - \frac{50(200)^2}{3(t+200)} \end{gathered} \tag{4-4}$$

Finally, the amount of salt in the container can be known when it is filled. Therefore, the elapsed time should also be known. *The amount of water increases 1 liter every minute* and initially it was *200 liters, 100 minutes* are the time necessary for filling the container in.

Several domain knowledge elements are identified: *container, mixture* (also called *solution*), *concentration of mixture,amount of water*, *amount of salt y(t)*, *capacity of container, entry velocity of salt.* Such elements are represented as classes and leaf concepts in the **structural view** of the PCS (see Figure **4-26**), which also allows for relating the tables of the data base.

Class *container has number*, and *capacity*, which are stored in a table of the data base (see Table **4-4**, the information context *capacity of container = 300 liters* is used). *Container* is also structurally related to the class *chemical expert* (class added as fictional information to real data in order to complete the simulation). *Chemical expert has code* and *name*, which has a table in the data base with his/her information (see Table **4-5**). *Container* is structurally related to the class *mixture* by using the triad *container has mixture. Mixture has code, liquid substance* (leaf concept added for representing the *water* and other liquid substances: *oil, solvent, alcohol, vinegar*), *soluble substance* (leaf concept added for representing the *salt* and other soluble substances: *sugar, sodium bicarbonate*), *liquid substance entry velocity, liquid substance exit velocity, soluble substance entry velocity, soluble substance exit velocity,* and *mininum velocity* (leaf concept used for representing the velocity of the mixture). *Mixture* is structurally related to the class *substance concentration* by using the triad *mixture has substance concentration* for representing the *concentration of mixture. Substance concentration has code, local time, liquid substance amount* (leaf concept for representing the *amount of water*), *liquid substance minimum amount* (leaf concept conformed by velocity and time in the same value of t in Equation 4-4. A implicit value in the context), and *soluble substance amount. Mixture* and *substance concentration* also have tables in the data base, which are stored in the dynamic view of the PCS.

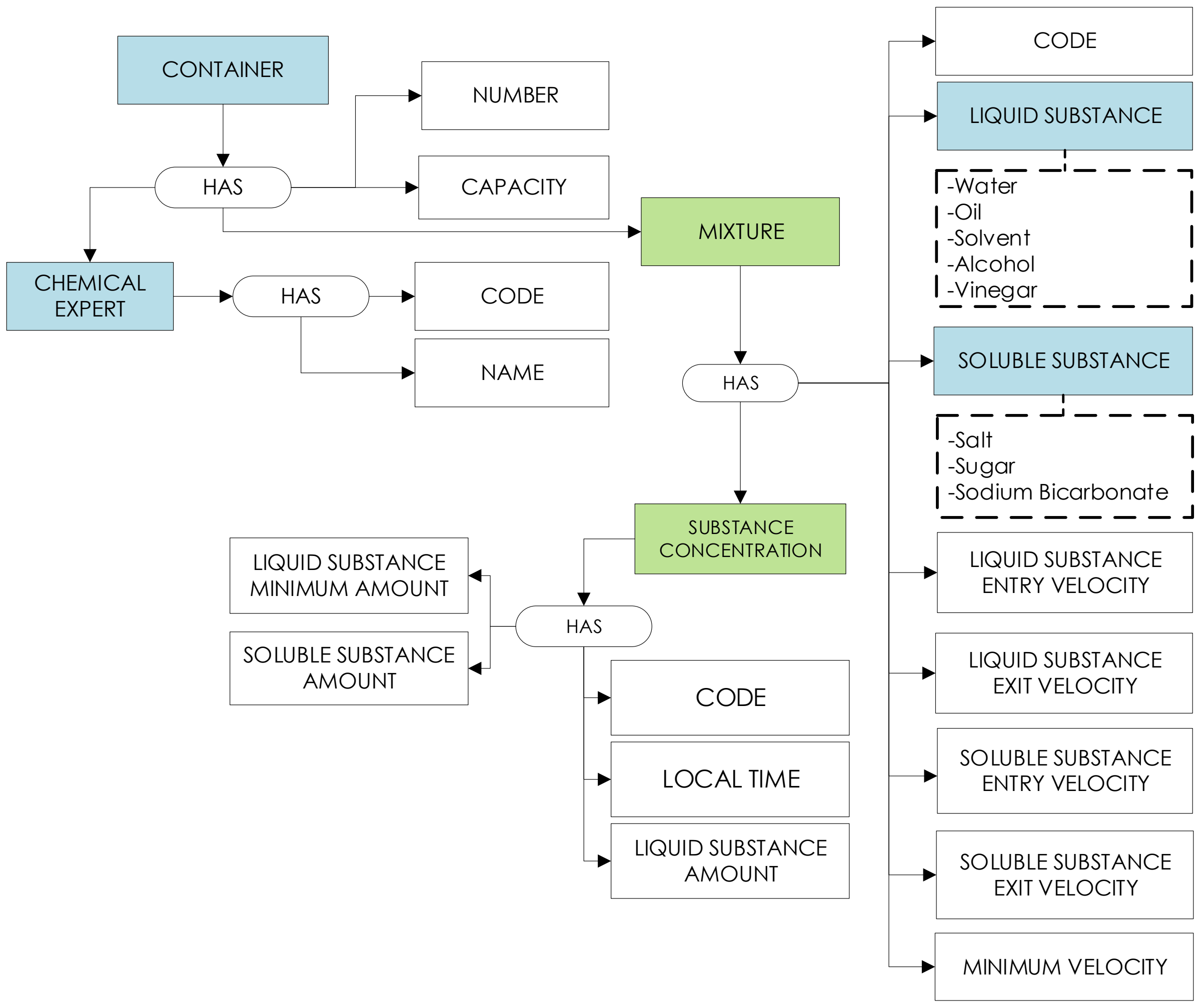


Figure 4-26 Structural view from PCS. The Authors based on Noreña *et al.* (2019)

Table 4-4 Container (Noreña *et al.*, 2019)

CONTAINER		
NUMBER	CAPACITY (liters)	CHEMICAL_EXPERT CODE
234	300	45

Table 4-5 Chemical Expert (Noreña *et al.*, 2019)

CHEMICAL_EXPERT	
CODE	NAME
45	Samir Guarín

Dynamic view from PCS comprises initial conditions, processes, and events. Initial conditions (see Figure **4-27**) are variable *time = 0 minutes*, parameter *simulation time = 360 minutes* (*time* and *simulation time* are added as fictional information to real data in order to complete the simulation in other possible mixtures), variable *mixture time = 0 minutes* (*t = 0 minutes*), parameters *mixture end time = 100 minutes*, *liquid substance initial amount* (*amount of water = 200 liters*), and *soluble substance initial amount* (*y(0) = 50 Kg of salt*).

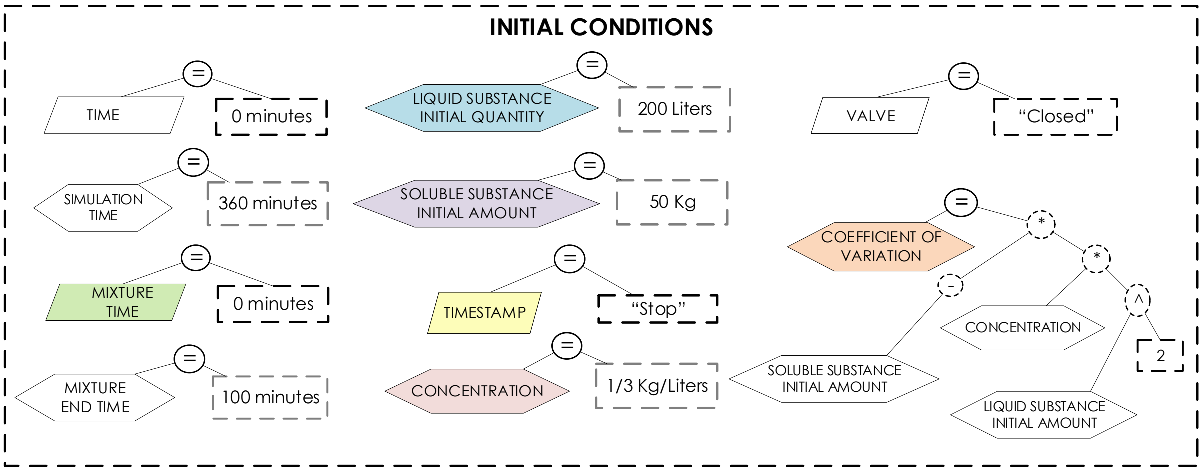


Figure 4-27 Initial conditions (Noreña *et al.*, 2019)

Timestamp = "stop" or "next" (variable added for controlling the time), *concentration = 1/3 Kg/Liters* (*concentration of mixture*), *valve = "open"* (and *"closed"* by inference), and coefficient of variation (which is obtained from Equation 4-4 and conceptually translated to

Equation 4-5).

Coefficient of variation with units is $C = \dfrac{-50Kg}{3liters}200^2 liters^2$ from Equation 4-4.

$$\begin{aligned} &\text{Translating to conceptual form: } Coefficient\ of\ variation = \\ &(-soluble\ substance\ initial\ amount\ * concentration)* \\ &(liquid\ substance\ initial\ amount)^2 \end{aligned} \tag{4-5}$$

Chemical expert inserts mixture is a process represented in the SSD (see Figure **4-28**). Before starting the simulation, *chemical expert selects container.number, mixture.liquid substance*, and *mixture.soluble substance* and *inserts mixture.liquid substance entry velocity* and *mixture.liquid substance exit velocity.*

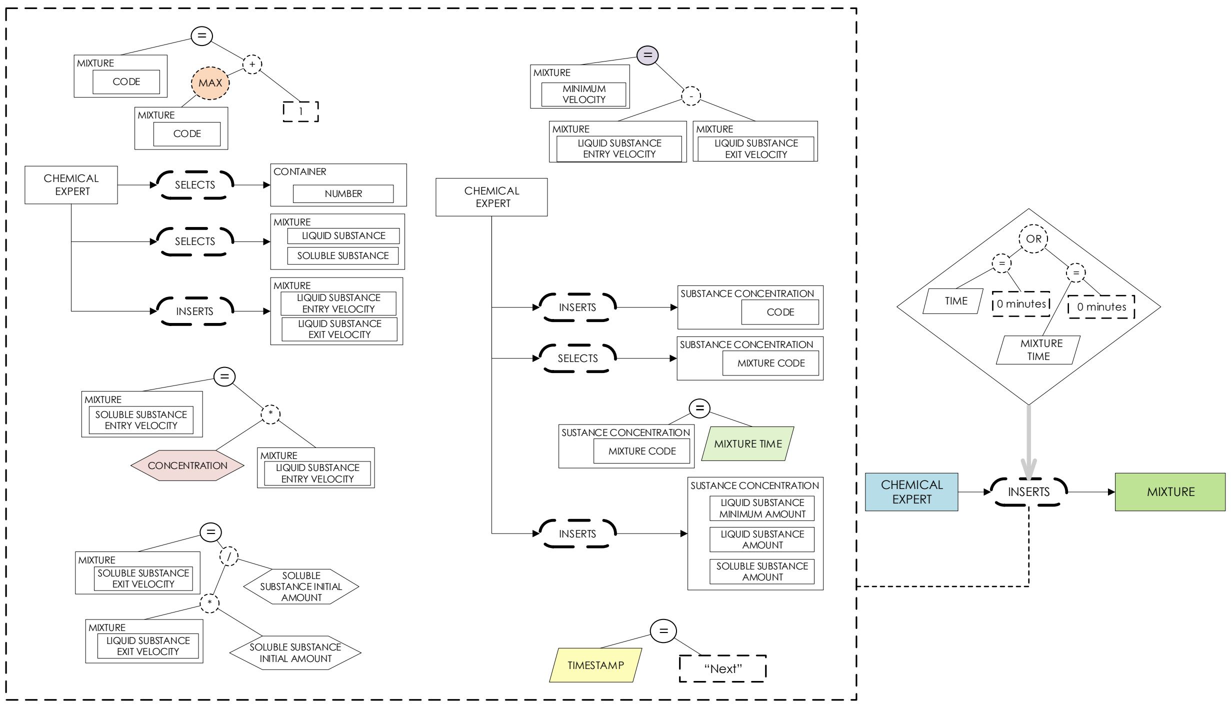


Figure 4-28 Process in chemical SSD (Noreña *et al.*, 2019)

Equations inside the specification of *chemical expert inserts mixture* are represented according to the context: *mixture.soluble substance entry velocity* (from "entry velocity of salt is *1/3 Kg/liters x 3 liters/minutes = 1Kg/minutes*"), *mixture.soluble substance exit velocity* (from *2y(t)/(t + 200 liters)*, it is *2 liters/min * 50 Kg / 200 liters*, *mixture.minimum velocity* (from "the reason for changing in every minute y'(t), it will be equal to the amount of salt entering to the container, minus the amount of salt coming out in the same minute", since the liquid substance contains the soluble substance). The results of such equations are derived attributes of the values selected and inserted by the *chemical expert.* Values used in the table *Mixture* (see Table **4-6**) are also related to the context: *liquid substance entry velocity = 3 liters/minutes, liquid substance exit velocity = 2 liters/minutes, soluble substance entry velocity = 1 Kg/minutes*. After, chemical expert inserts the first registry of the Table **4-7** by using the actions *chemical expert inserts substance concentration, selects substance concentration.mixture code, substance concentration.local time = mixture time*, and *inserts liquid substance minimum amount, liquid substance amount*, and *soluble substance amount.* Finally, *timestamp = "next"* for starting the simulation.

Table 4-6 Mixture (Noreña *et al.*, 2019)

MIXTURE								
CODE	CONTAINER NUMBER	LIQUID SUBSTANCE	SOLUBLE SUBSTANCE	LIQUID SUBSTANCE ENTRY_VELOCITY (liters/minutes)	SOLUBLE SUBSTANCE ENTRY_VELOCITY (Kg/minutes)	LIQUID SUBSTANCE EXIT_VELOCITY (liters/minutes)	SOLUBLE SUBSTANCE EXIT_VELOCITY (Kg/minutes)	MINIMUM VELOCITY (liters/minutes)
1	234	Water	Salt	3	1	2	½	1

Time passes, mixture starts, mixture ends (see Figure **4-29**, **4-30**, and **4-31**), and *substance concentration increases* (see Figure **4-32**) are events represented in chemical SSD. *Time passes* is the event used for controlling the *simulation time* and *mixture time* (see Figure **4-29**). *Time* increases every minute according to the context. Several mixtures can be performed in the *simulation time.*

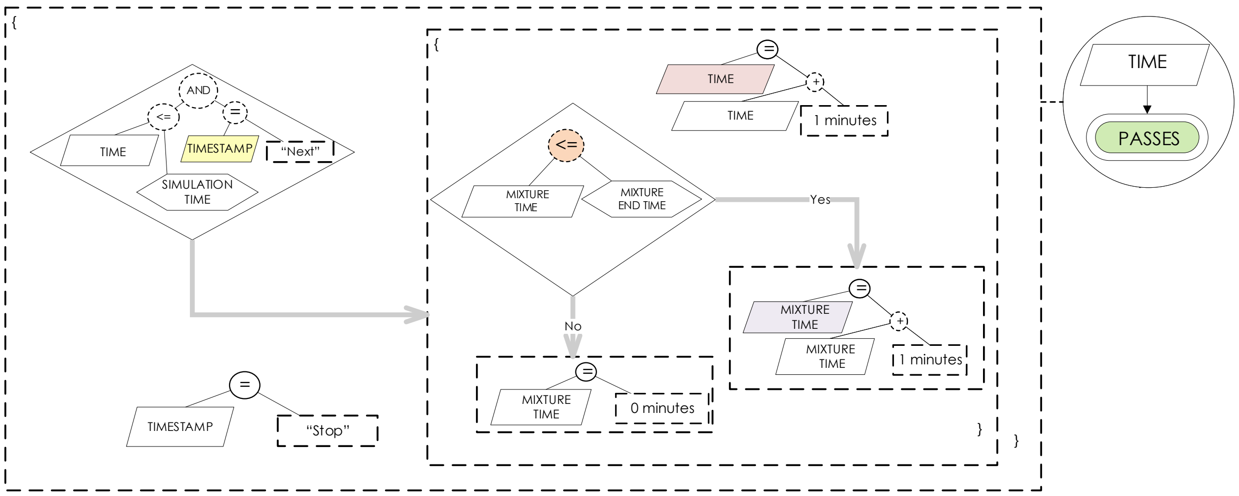


Figure 4-29 Event: time passes. The Authors based on Noreña *et al.* (2019)

Mixture starts is the event used for opening the *valve* related to the elements of the condition: *mixture time, container.capacity*, and *timestamp* in "next" (See Figure **4-30**).

Mixture ends is the event added for closing the *valve* related to the elements of the condition: *mixture time* and *time* (See Figure **4-31**).

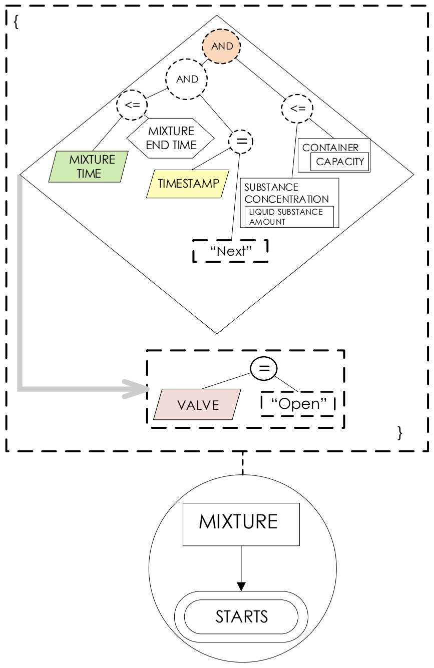


Figure 4-30 Event: mixture starts. The Authors based on Noreña *et al.* (2019)

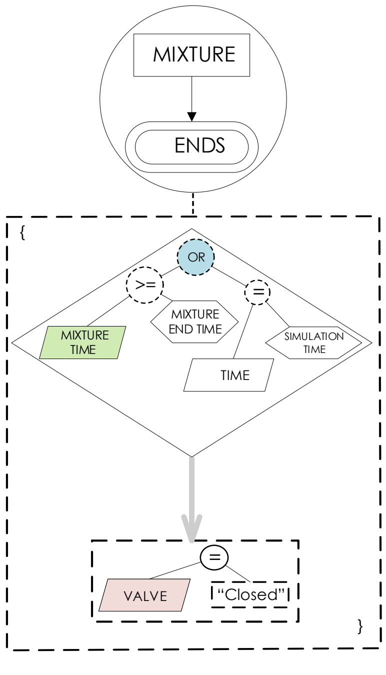


Figure 4-31 Event: mixture ends. The Authors based on Noreña *et al.* (2019)

Substance concentration increases (see Figure **4-32**) is the event for automatically inserting the values of the class *susbstance concentration* in Table **4-7**. Such values are *code, mixture code, local time*, *liquid substance minimum amount* (from the value conformed by velocity and time in the same value of *t* in Equation 4-4), *liquid substance amount* (from the context: *the amount of water increases 1 liter every minute*), and *soluble substance amount* (from Equation 4-6). Variable *timestamp* is used for stopping the time when a value is inserted.

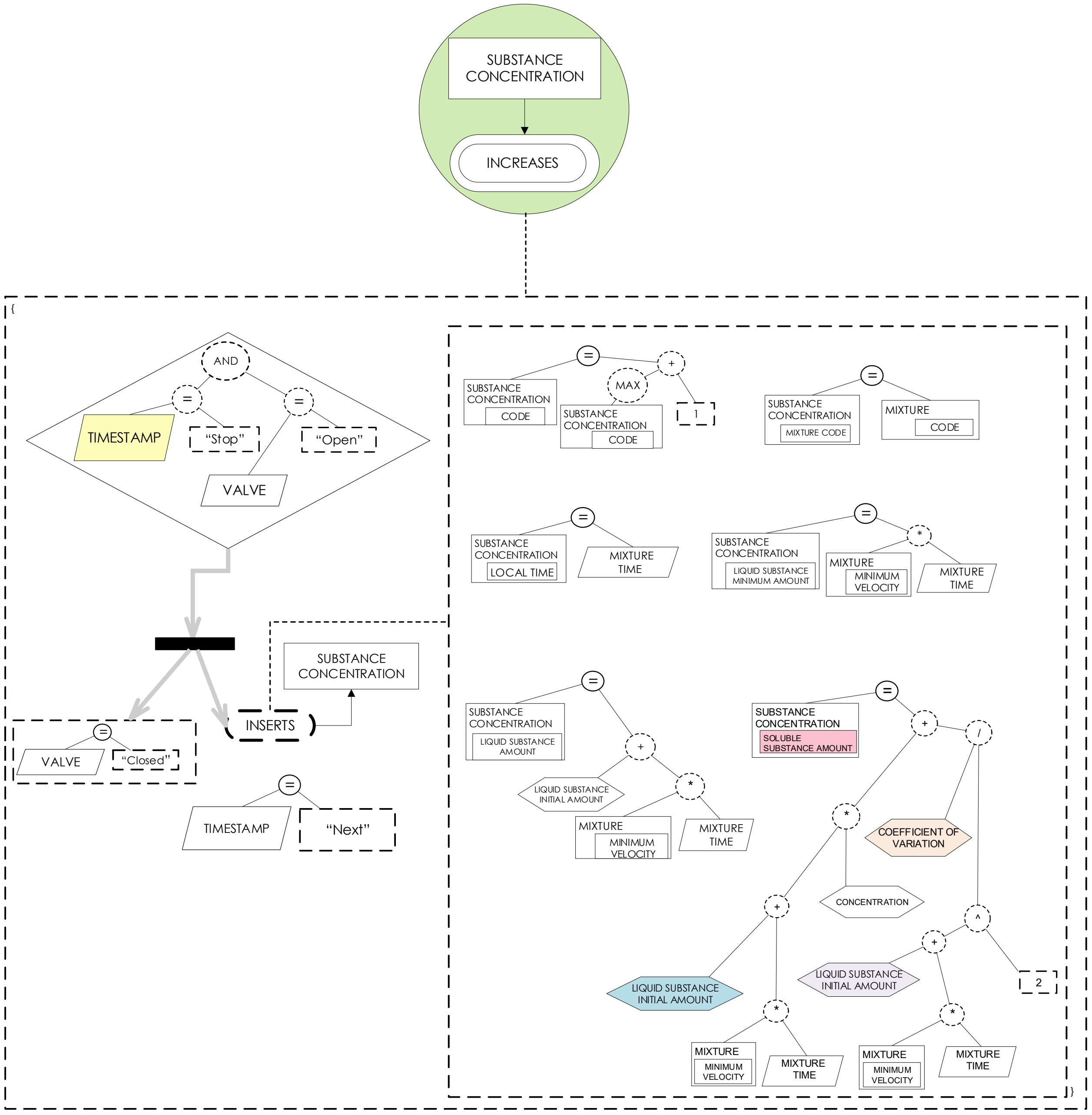

SUBSTANCE CONCENTRATION
INCREASES
AND
TIMESTAMP
"Stop"
"Open"
VALVE
SUBSTANCE CONCENTRATION
INSERTS
VALVE
"Closed"
TIMESTAMP
"Next"
SUBSTANCE CONCENTRATION CODE
MAX
SUBSTANCE CONCENTRATION CODE
1
SUBSTANCE CONCENTRATION MIXTURE CODE
MIXTURE CODE
SUBSTANCE CONCENTRATION LOCAL TIME
MIXTURE TIME
SUBSTANCE CONCENTRATION LIQUID SUBSTANCE MINIMUM AMOUNT
MIXTURE MINIMUM VELOCITY
MIXTURE TIME
SUBSTANCE CONCENTRATION LIQUID SUBSTANCE AMOUNT
LIQUID SUBSTANCE INITIAL AMOUNT
MIXTURE MINIMUM VELOCITY
MIXTURE TIME
SUBSTANCE CONCENTRATION SOLUBLE SUBSTANCE AMOUNT
COEFFICIENT OF VARIATION
CONCENTRATION
LIQUID SUBSTANCE INITIAL AMOUNT
LIQUID SUBSTANCE INITIAL AMOUNT
2
MIXTURE MINIMUM VELOCITY
MIXTURE TIME
MIXTURE MINIMUM VELOCITY
MIXTURE TIME

Figure 4-32 Event: substance concentration increases. The Authors based on Noreña *et al.* (2019)

Table 4-7 Substance concentration. The Authors based on Noreña *et al.* (2019)

SUBSTANCE_CONCENTRATION					
CODE	**MIXTURE CODE**	**LOCAL TIME** (minutes)	**LIQUID SUBSTANCE MINIMUM AMOUNT** (liters)	**LIQUID SUBSTANCE AMOUNT** (liters)	**SOLUBLE SUBSTANCE AMOUNT** (Kg)
500	1	0	0	200	50
501	1	1	1	201	50,4
502	1	2	2	202	51
…600	1	…100	…100	…300	…92,6

$$\begin{aligned} &\textit{Soluble substance amount} \text{ with units is } y(t) = \\ &\frac{(200liters + t\,minutes * liters/minutes)1Kg}{3liters} \\ &+\frac{-50Kg(200liters)^2}{3liters(200liters + t\,minutes * liters/minutes)} \\ &\text{from Equation 4-4.} \end{aligned} \tag{4-6}$$

$$\begin{aligned} &\text{Translating to conceptual form: } Soluble\ substance\ amount = \\ &liquid\ substance\ initial\ amount\ + mixture.minimum\ velocity * mixture\ time \\ &*concentration \\ &+\frac{coefficient\ of\ variation}{(liquid\ substance\ initial\ amount\ + mixture.minimum\ velocity * mixture\ time)^2} \end{aligned}$$

We show the complete PCS in Figure **4-33**, which includes the flow of the whole system guided by the *implication* link, which start when a conditional event triggers the process *chemical expert inserts mixture*. After, the events *time passes* and *mixture starts* trigger the event *substance concentration increases*. Finally, *mixture ends* in the indicated time.

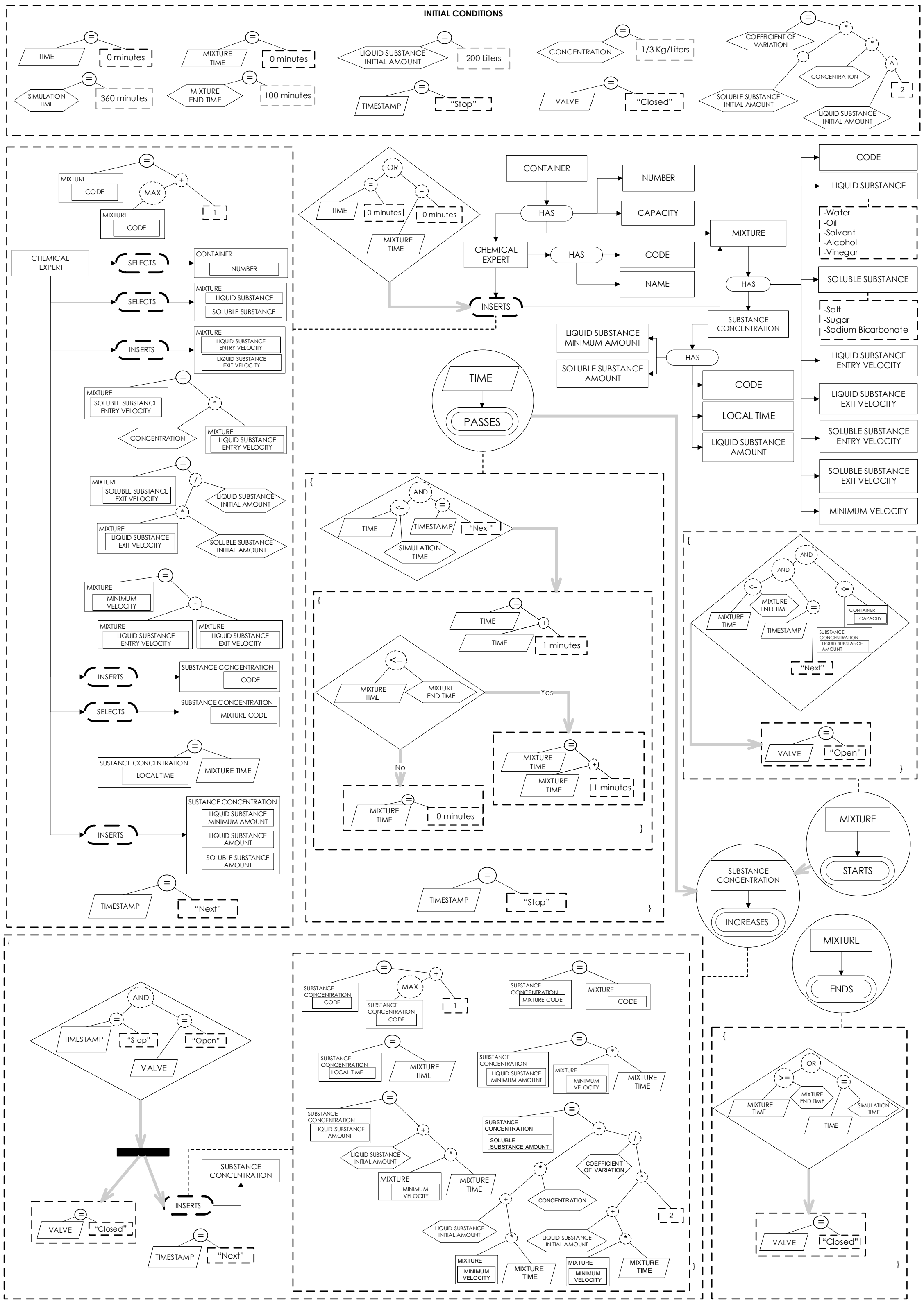


Figure 4-33 Events in chemical SSD. The Authors translated from Noreña *et al.* (2019)

4.5. Events Represented in PCS

We summarize the events represented by using the extended PCS in Table **4-8**.

Table 4-8 Events represented in PCS. The Authors

Event	Scientific software domain	Reference
Population value increases	Statistics	Calle, Noreña, & Zapata, in process
Mixture starts	Chemistry	Noreña, Zapata, & Villamizar, 2019
Substance concentration increases		
Earthquake arrives	Seismology (real study)	Noreña & Zapata, in process
Volcano erupts	Geology, Metereology	Noreña *et al.*, 2018
Well pression increases	Petroleum Engineering	Velásquez, 2019
Epidemy increases, patient quantity grows	Medicine	Noreña & Zapata, 2018b; Noreña, 2018; Zapata *et al.*, in process
Environmental noise emerges	Environmental engineering (real study)	Durango, Noreña, & Zapata, 2018
Bacteria quantity grows	Biology	Noreña & Zapata, 2018 (poster)
Freezing happens	Simulation	Noreña & Zapata, 2019
Call starts Signal emerges (deterministic and random signal events)	Electronic	Noreña & Zapata, 2018a
Seafood arrives to warehouse (monitoring system) Message event appears Temperature changes	Industrial domain	Noreña *et al.*, in process
Sensor starts	Environmental engineering	Noreña & Zapata, 2018c
Time passes	All	All
Thersmistor measure starts	Automatic systems (lab study)	Tutoring Students
Production volumen arises Inflaction rate increases, decreases Well blowout ocurrs victim suffers violent death	Economy Petroleum Engineering Legal medicine	Students in Requirements engineering course

5 Validation

The universe must not be narrowed down to the limit of our understanding, but our understanding must be stretched and enlarged for taking in the image of the universe as it is discovered.
—Francis Bacon

5.1. Experimental Validation

5.1.1. Planning Experiment

We select the experimental validation method: *expert opinion*; the design of a PCS is submitted to a panel of experts, who should understand how such a model interacts with problem domains. Such a method is supported by the *design science methodology for information systems and software engineering* (Wieringa, 2014). We select the experimental process supported by the *experimentation in software engineering* (Wohlin *et al.*, 2012).

Table 5-1 Experiment planification. The Authors

Goal	Hypotheses	Variables		Questions
Analyze *the extended PCS* for the purpose of *evaluation* with respect to *understandability and usability* from the point of view of *scientists and software analysts* in the context of *students, professors, and professionals in the area*		Independent	Expert profile (professor, professional in the area, and student)/area/experience years range	Profile type and performance area/ experience years
	$H1_0$. the PCS is unintelligible *$H1_1$*. the PCS is understandable	Dependent	Domain/ PCS understandability	Q1. Could you understand the PCS?
	$H2_0$. the event and mathematical notation is unintelligible *$H2_1$*. the Event and mathematical notation is understandable		Event/mathematical notation understandability	Q2. Could you understand the events/mathematical notation?
	$H3_0$. The PCS is unusable for representing events in SSD *$H3_1$*. The PCS can be used for representing events in SSD		PCS Usability	Q3. Could the PCS be used in SSD?
	S. Supported R. Refuted			

Hypotheses, research questions, and *variables* are selected in Table **5-1**. *Experimental design* is especially quantitive but also it has qualitative aspects.We use statistical analysis with the techniques of mean (average), median (central value), mode (most repeated value), standard deviation (SD, fluctuation arithmetic average from the mean), coefficient of variation (CV, relationship between the size of the mean and the variability of the variable), interquartile range (IQR, measure of variability when the measure of central position is the median, which results of *higher quartile Q - lower Q*, such opinions are selected for quantity and experience years), and average percent of majority opinions (APMO, from Equation 5-1) for a scale of 5 points (Likert). Chi-squared of Pearson (hypothesis test for comparing the observed distribution to an expected distribution of the data), contingency coefficient (CC, relationship between two and more variables), and frequency for a escale of two points (nominal). Consensus criteria for a significance level are defined according to the measures indicated for questionnaires (Heiko, 2012; Holey *et al.*, 2007), which are presented in every statistical analysis. Results are analized from the SPSS Statistics program (except the APMO).

$$APMO = \frac{majority\, agreements + majority\, disagreements}{total\, opinion\, expressed} x\, 100\,\% \tag{5-1}$$

5.1.2. Executing and Analyzing Experiment

Experiment data and report are generated from a survey (descriptive research instrument selected for collecting data) performed to scientific, computing, and simulation experts.

Scientific Experts

We carry out an experiment by using the PCS presented to chemical SSD in the lab study (see Figure **4-33**). Such an experiment allows for analyzing the level of understanding of chemical experts about the proposed model. Sample size is 36 experts (professionals in the area, professors, and students) from universities with programs in chemistry and companies in the chemical field in Colombia within 30 days (see Table **5-2**).

Table 5-2 Experiment sample (Noreña *et al.*, 2019)

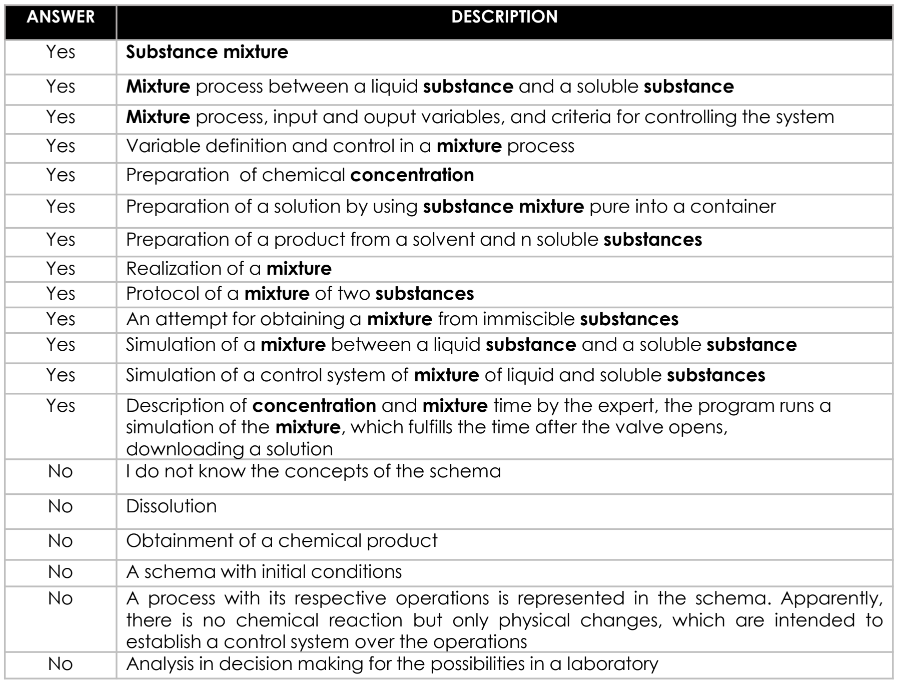
SAMPLE			
EXPERT PROFILE	**QUANTITY**	**PERCENTAGE**	**EXPERIENCE YEARS RANGE**
Professional In the area	18	50%	2-18
Professor	8	22%	3-18
Student	10	28%	0-3
TOTAL	**36**	**100%**	**0-18**

Experiment is performed in three steps:

Intutive recognition of the PCS, the chemical experts have not prior knowledge of the PCS notation (both the notation in Figure **2-1** and the extension to PCS). Pre-conceptual schema of the lab study (see Figure **4-33**) is presented without explanation of the context for achieving an intuitive recognition, which consists of understanding the domain and its elements by using the PCS notation (Noreña *et al.*, 2019).

Description of the domain, the chemical experts interpret the chemical processes and events represented in the PCS and textually and qualitatively describe them in the survey by using the question: please relate in your own words what is the process/theme/approach represented in the pre-conceptual schema? corresponding to the variable *domain*. We translate the performed description to answers in nominal dichotomous scale *Yes/No* for validating the description of the domain. *Yes* is the answer used for descriptions, including the concepts: *mixture, substance*, and *concentration* for the specific affirmation *it is a mixture process*. *No* is the answer used for all descriptions different according to such an affirmation (Noreña *et al.*, 2019). Some descriptions and answers are presented in Table **5-3**.

Table 5-3 Description of the domain. The Authors based on (Noreña *et al.*, 2019)



ANSWER	DESCRIPTION
Yes	**Substance mixture**
Yes	**Mixture** process between a liquid **substance** and a soluble **substance**
Yes	**Mixture** process, input and ouput variables, and criteria for controlling the system
Yes	Variable definition and control in a **mixture** process
Yes	Preparation of chemical **concentration**
Yes	Preparation of a solution by using **substance mixture** pure into a container
Yes	Preparation of a product from a solvent and n soluble **substances**
Yes	Realization of a **mixture**
Yes	Protocol of a **mixture** of two **substances**
Yes	An attempt for obtaining a **mixture** from immiscible **substances**
Yes	Simulation of a **mixture** between a liquid **substance** and a soluble **substance**
Yes	Simulation of a control system of **mixture** of liquid and soluble **substances**
Yes	Description of **concentration** and **mixture** time by the expert, the program runs a simulation of the **mixture**, which fulfills the time after the valve opens, downloading a solution
No	I do not know the concepts of the schema
No	Dissolution
No	Obtainment of a chemical product
No	A schema with initial conditions
No	A process with its respective operations is represented in the schema. Apparently, there is no chemical reaction but only physical changes, which are intended to establish a control system over the operations
No	Analysis in decision making for the possibilities in a laboratory

We use the techniques selected for a scale of two points, which are used for statistically analyzing the answers in the description of the domain. Obtained results for the variable *domain* (see Table **5-4**) are at a significance level according to the consensus criteria. Since, the value of chi-squared of Pearson is in an acceptation zone, the value of CC presents a correlation between experts and the description of the domain, and the value of the frequency is 81 % for the answer *yes* (See Figure **5-1**) in the concepts used in the description. Such values indicate the experts understand the model without prior knowledge of the domain and the PCS notation. Results contribute to support the hypothesis $H1_1$ (see Table **5-1**).

Table 5-4 Statistical analysis of the domain. The Authors

STATISTICAL ANALYSIS			
VARIABLE		Domain	CONSENSUS CRITERIA
N	Valid	36	
	Lost	0,0	
Chi-squared of Pearson		3,719	<5
Contingency Coefficient		0,306	<1
Frequency		81%	>67% (Significance)
Hypothesis		$H1_1$: S	

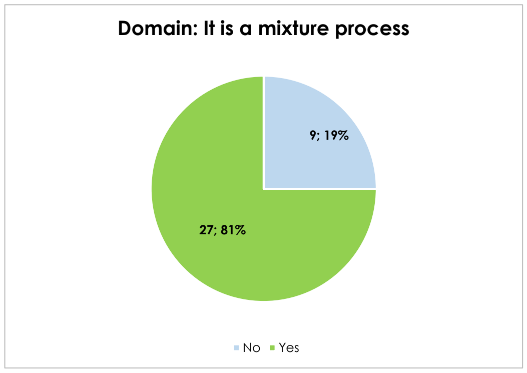


Figure 5-1 Frequency of the description of the domain. The Authors

Evaluation of the PCS, the chemical experts evaluate the model. Such an evaluation is performed by using three questions: (i) evaluating the understanding level of PCS (Likert scale: from 1 to 5, 1 is the lowest value and 5 the highest) corresponding to the variable *PCS understanding*, (ii) could you understand the mathematical notation? corresponding to the variable *mathematical notation understandability*, and (iii) do you consider as an expert in your domain that the PCS can be usable for understanding events/phenomena and processes, *e.g.*, in a simulation/software development process? corresponding to the variable *PCS usability* (Likert scale: strongly disagree, disagree, neutral, agree, strongly agree; see Figure **5-2**). We use the selected techniques for a scale of 5 points, which are used for statistically analyzing the answers in the evaluation to PCS (see Table **5-5**).

Most chemical experts (25) evaluate the *PCS understandability* between assessment levels 3 and 5 (see *PCS understandability* in Figure **5-2**), for indicating the PCS is understandable without prior knowledge of the PCS and their notation. The same frequency of professionals in the area (5), can be observed for the three assessment levels 1, 3, and 4. However, most answers is at levels 3 and 4 with a frequency of 10, to this value is added 1 at level 5, indicating a majority. Most professors evaluate such a variable in the level 4.

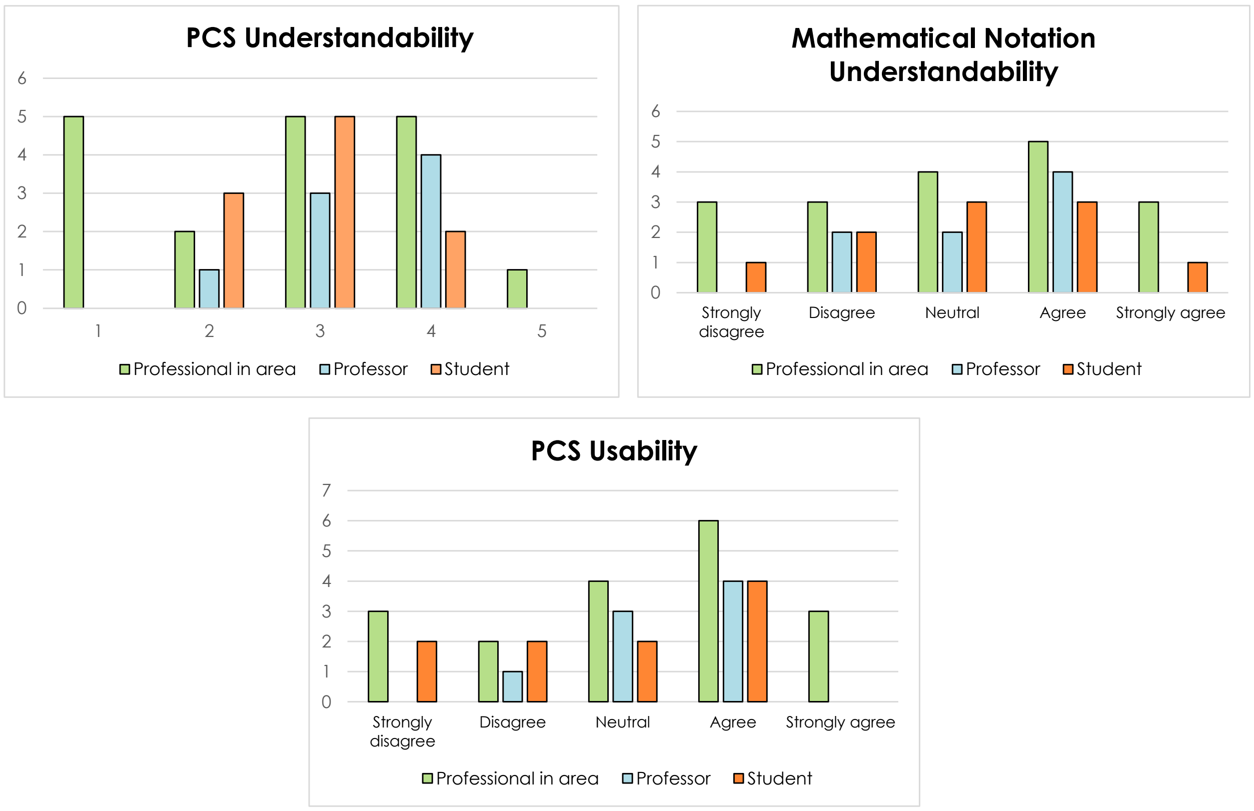


Figure 5-2 Frequency of the PCS evaluation. The Authors based on Noreña *et al.* (2019)

Three profile experts also *agree* on the mathematical notation is understandable and the PCS can be usable for understanding events/phenomena and processes in a simulation/software development process (see *mathematical notation understandability* and *PCS usability* in Figure **5-2**).

Obtained results (see Table **5-5**) in the mean, median, mode, SD, CV, IQR, and APMO (see Equations 5-2) indicate a significance level according to the consensus criteria, which contribute to support the hypothesess $H1_1$, $H2_1$, and $H3_1$ (see Table **5-1**).

$$\begin{aligned} APMO &= \frac{25+11}{36} x\, 100\,\% = 80,4\,\% \quad \text{for PCS understandability} \\ APMO &= \frac{17+10}{36} x\, 100\,\% = 75,0\,\% \quad \text{for Mathematical Notation understandability} \\ APMO &= \frac{16+11}{36} x\, 100\,\% = 75,0\,\% \quad \text{for PCS usability} \end{aligned} \tag{5-2}$$

Table 5-5 Statistical analysis of the PCS evaluation. The Authors

STATISTICAL ANALYSIS					
VARIABLES		PCS Understandability	Mathematical Notation Understandability	PCS Usability	CONSENSUS CRITERIA
N	Valid	36	36	36	
	Lost	0,0	0,0	0,0	
Mean		3,0	3,1	3,1	<=5
Median		3,0	3,0	3,0	<=5
Mode		3,0	4,0	4,0	>=3
Standard Deviation		1,1	1,2	1,2	±1,0
Coefficient of Variation		0,4	0,4	0,4	<=0,5
Quartile	Lower	3,0	3,0	3,0	>=3
	Higher	4,0	4,0	4,0	5
Interquartile range IQR		1,0	1,0	1,0	<=1
Average percent of majority opinions APMO		80,4	75,0	75, 0	>69,7%
Hypothesis		$H1_1$: S	$H2_1$: S	$H3_1$: S	

Computing and Simulation Experts

We carry out another experiment by using several PCS in SSD *e.g.*, electronic (Noreña & Zapata, 2018a), geology (Durango *et al.*, 2018), epidemiology, (Noreña & Zapata, 2018b; Noreña, 2018), and industry (Noreña & Zapata, 2019). Such an experiment allows for analyzing the level of understanding of computing experts (19) and simulation experts (20) from international universities and companies in computational sciences and simulation about the proposed model. The experiment is performed in two steps:

Recognition of the PCS, the experts have an explanation of the PCS notation (both the notation in Figure **2-1** and the extension to PCS) and the SSD representation.

Evaluation of the PCS, the experts evaluate the model. Such an evaluation is performed by using three questions: (i) Could you understand the PCS? corresponding to the variable *PCS understanding*, (ii) Could you understand the event notation? corresponding to the variable *event notation understandability*, and (iii) do you consider as an expert in your domain PCS can be usable in SSD? corresponding to the variable *PCS usability*, with answers in nominal dichotomous scale *Yes/No* and a qualitatively answer in every question *Why?* for extending their answers.

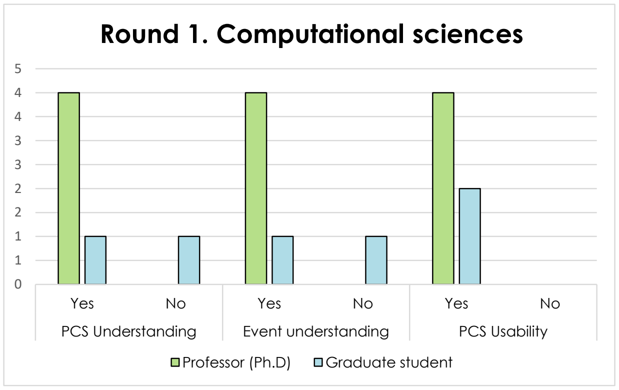
Experiment to *computing experts* (professors, students, and professional in the area) is developed in two rounds. The sample for such rounds is presented in Table **5-6**.

Table 5-6 Sample of computational sciences. The Authors

SAMPLE (Round 1. Computational sciences)			
EXPERT PROFILE	QUANTITY	PERCENTAGE	PLACE
Professor (Ph.D.)	4	66,6%	Argentina Spain Paraguay
Ph.D. student	2	33, 4%	Argentina Colombia
TOTAL	**6**	**100%**	

SAMPLE (Round 2. Computational sciences)			
EXPERT PROFILE	QUANTITY	PERCENTAGE	PLACE
Professor (Ph.D.)	3	23, 1%	Mexico
Undergraduate student	4	30, 7%	
Graduate student	3	23, 1%	
Professional in the area	3	23, 1%	
TOTAL	**13**	**100%**	

We use the techniques selected for a scale of two points, which are used for statistically analyzing the answers. Some qualitative answers are presented in Table **5-7**. Most experts confirm the variables *PCS understandability, event understandability*, and *PCS usability* (see Figure **5-3** and Tables **5-8** and **5-9**).



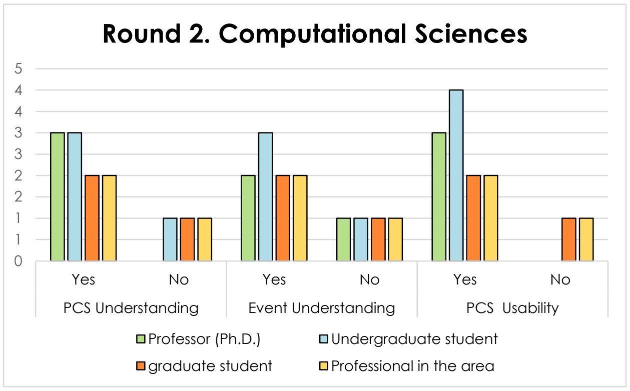


Figure 5-3 Frequency Round 1 and 2 of computational sciences. The Authors

Table 5-7 Qualitative answers of computational sciences. The Authors

Questions	Qualitative Answers
Could you understand the PCS? Why?	PCS allowed me to perceive the understanding of the logic-model (professor Ph.D.)
	PCS is easy to visualize (professor Ph.D.)
	Not sure (professor Ph.D.)
	I do not know the meaning pre-conceptual schema (professor Ph.D.)
Could you understand the event notation? Why?	PCS allowed me to understand information structures embbed in the events
	I can see results (professor Ph.D.)
Could the PCS be used in SSD? Why?	PCS can be applied to any decision-making or information systems
	Seems useful (professor Ph.D.)
	It is a visualization of a model/algorithm
	Not sure (professor Ph.D.)

Table 5-8 Statistical Analysis: Round 1 of computational sciences. The Authors

STATISTICAL ANALYSIS					
VARIABLE		PCS Understanding	Event Understanding	PCS Usability	CONSENSUS CRITERIA
N	Valid	6	6	6	
	Lost	0,0	0,0	0,0	
Chi-squared of Pearson		2,4	2,4	0,0	<5
Contingency Coefficient		0,535	0, 535	0,0	<1
Frequency		(5) 83,3%	(5) 83,3%	(6) 100%	>67% (Significance)
Hypothesis		$H1_1$: S	$H2_1$: S	$H3_1$: S	

Table 5-9 Statistical Analysis: Round 2 of computational sciences. The Authors

STATISTICAL ANALYSIS					
VARIABLE		PCS Understanding	Event Understanding	PCS Usability	CONSENSUS CRITERIA
N	Valid	13	13	13	
	Lost	0,0	0,0	0,0	
Chi-squared of Pearson		1,264	0,90	2,758	<5
Contingency Coefficient		0,298	0, 083	0,418	<1
Frequency		(10) 76,9%	(9) 69,3%	(11) 84, 6 %	>67% (Significance)
Hypothesis		$H1_1$: S	$H2_1$: S	$H3_1$: S	

Experiment of *simulation experts* (professors and professional in the area) is developed in two rounds. Sample for such rounds is presented in Table **5-10**.

Table 5-10 Sample of simulation. The Authors

SAMPLE (Round 1. Simulation)			
EXPERT PROFILE	QUANTITY	PERCENTAGE	PLACE
Professional in the area	2	18,2%	Canada, China
Professor (Ph.D.)	9	81, 8%	US
TOTAL	**11**	**100%**	

SAMPLE (Round 2. Simulation)			
EXPERT PROFILE	QUANTITY	PERCENTAGE	PLACE
Professional in the area	2	22,3%	US
Professor (Ph.D.)	7	77, 7%	
TOTAL	**9**	**100%**	

We use the techniques selected for a scale of two points, which are used for statistically analyzing the answers. Some qualitative answers to the questions are presented in Table **5-11**. Most experts confirm the variables *PCS understandability, event understandability*, and *PCS usability* (see Figure **5-4** and Tables **5-12** and **5-13**).

Table 5-11 Qualitative answers to simulation. The Authors

Questions	Qualitative Answers
Could you understand the PCS? Why?	It is very clear (Professional in the area)
	It is very interesting (graduate student)
Could you understand the event notation? Why?	Figures used allows for understanding it (Professional in the area)
	Concepts are unknown (undergraduate student and professional in the area)
Could the PCS be used in SSD? Why?	The component extension allows for extending the capability for representing mathematical components (Professional in the area)

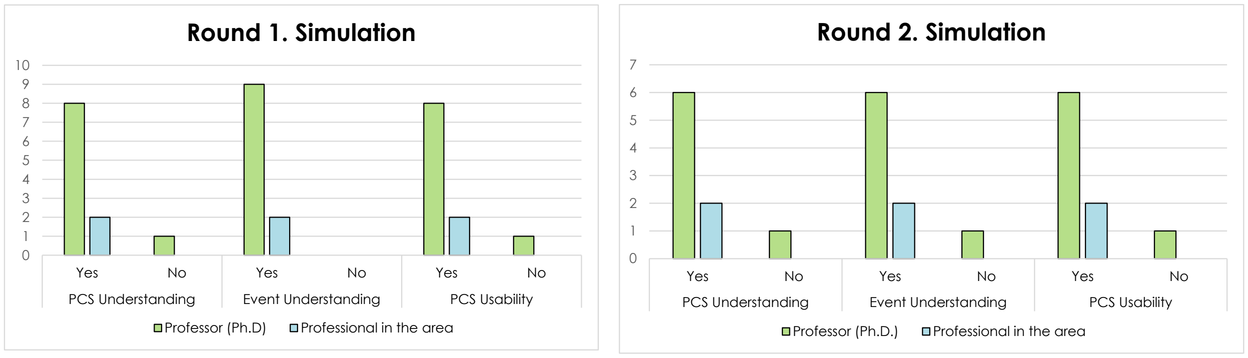


Figure 5-4 Frequency Round 1 and 2 to Simulation. The Authors

Table 5-12 Statistical Analysis: Round 1 of simulation. The Authors

STATISTICAL ANALYSIS					
VARIABLE		PCS Understanding	Event Understanding	PCS Usability	CONSENSUS CRITERIA
N	Valid	11	11	11	
	Lost	0,0	0,0	0,0	
Chi-squared of Pearson		2,44	0,0	2,44	<5
Contingency Coefficient		0,147	0,0	0,147	<1
Frequency		(10) 91%	(11) 100%	(10) 91%	>67% (Significance)
Hypothesis		$H1_1$: S	$H2_1$: S	$H3_1$: S	

Obtained results to both computing and simulation experts are at a significance level (answers *Yes*) according to the consensus criteria. Since, the value of chi-squared of Pearson in an acceptation zone, the value of CC presenting a correlation between the experts and the PCS understandability (see Figure **5-1**), and the frequency indicate the experts understand the model and the PCS notation. Results contribute to support the hypotheses $H1_1$, $H2_1$, and $H3_1$ (see Tables **5-8**, **5-9**, **5-12**, and **5-13**).

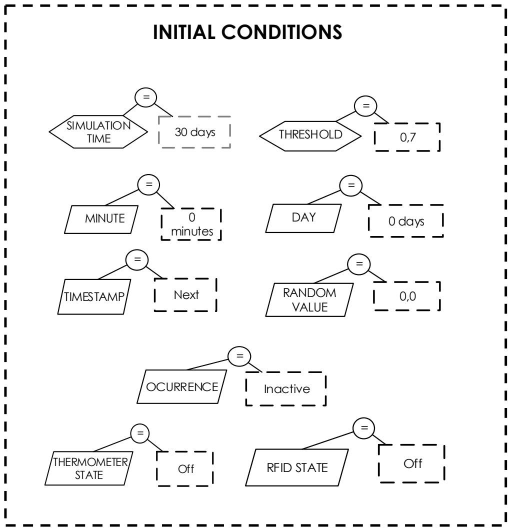
Table 5-13 Statistical Analysis: Round 2 of simulation. The Authors

STATISTICAL ANALYSIS					
VARIABLE		PCS Understanding	Event Understanding	PCS Usability	CONSENSUS CRITERIA
N	Valid	9	9	9	
	Lost	0,0	0,0	0,0	
Chi-squared of Pearson		0,321	0,321	0,321	<5
Contingency Coefficient		0,186	0,186	0,186	<1
Frequency		(8) 88,8%	(8) 88,8%	(8) 88,8%	>67% (Significance)
Hypothesis		$H1_1$:S	$H2_1$: S	$H3_1$: S	

5.2. Software Application

Python Code

A model is defined by using PCS for representing applications of complex event processing.



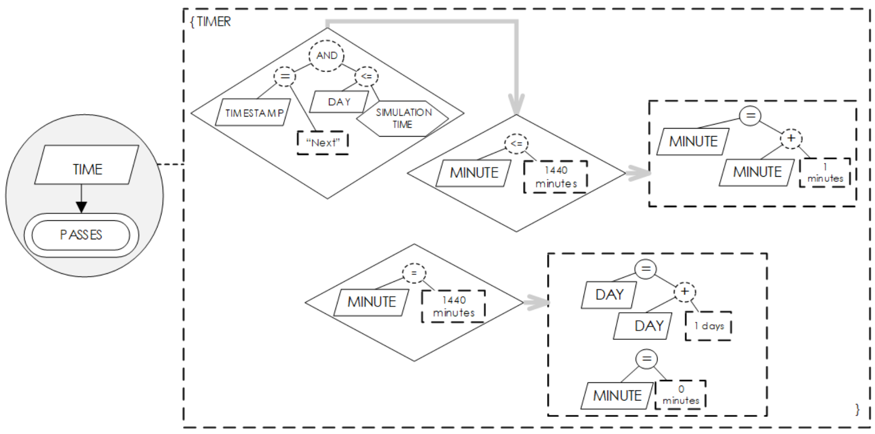
```
#CONSTANTS - INITIAL CONDITIONS

SIMULATION_TIME = 30 #Measured in days
THRESHOLD = 0.7 #Probability in [0 - 1] range

#INDEPENDENT VARIABLES - INITIAL CONDITIONS
MINUTE = 0 #measured in minutes
DAY = 0 #measured in days
TIMESTAMP = "Next" #Stop or Next
RANDOM_VALUE = 0.0 #in [0 - 1] range
OCURRENCE = "Inactive" #Active or Inactive
THERMOMETER_STATE = "Off" #On or Off
RFID_STATE = "Off" #On or Off
```

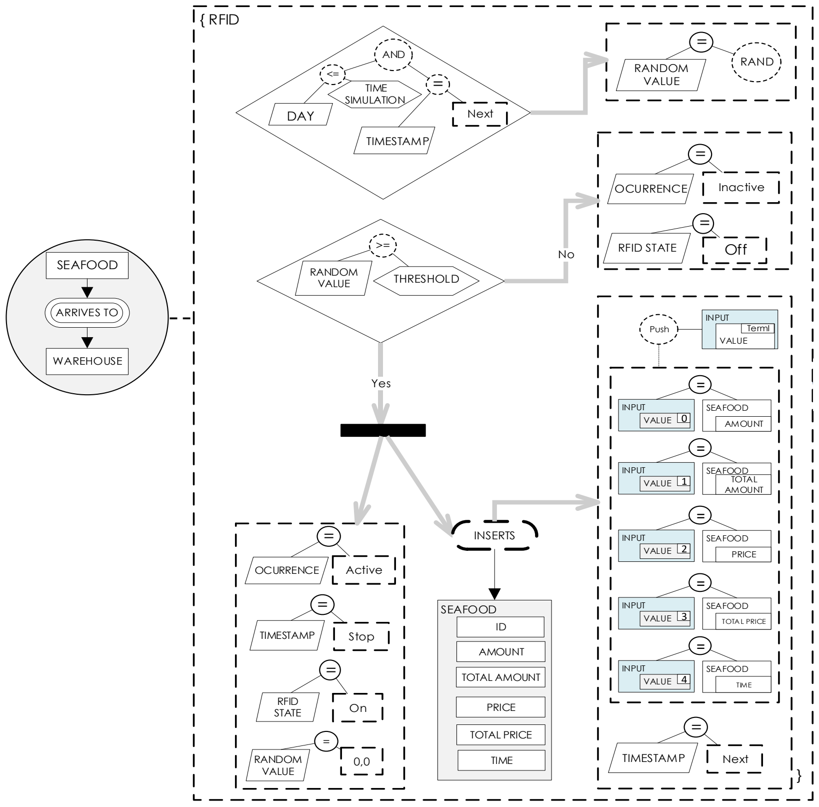
Figure 5-5 Initial conditions to Python code (Noreña *et al.*, in process)

Such a model is applied to a monitoring system of seafood in a warehouse in the industrial domain (Noreña & Zapata, 2019). Such PCS is translated into Python code (see Figures **5-5**, **5-6**, and **5-7**) and is also simulated in Python (in 30 days, see Figure **5-8**, the events are filtered and published to subscribers). The model is performed during the internship at the University of Toronto. A part of the applied PCS is presented in Figure **5-9**.



```
def TIME_PASSES():
  global MINUTE
  global DAY
  global TIMESTAMP
 if(DAY <= SIMULATION_TIME
    and
    TIMESTAMP == "Next"):
     #One minute passed
     if(MINUTE<1440):
       MINUTE = MINUTE + 1
 if(MINUTE == 1440):
     #Next day begins
     DAY = DAY + 1
     MINUTE = 0
```

Figure 5-6 Timer to Python code (Noreña *et al.*, in process)

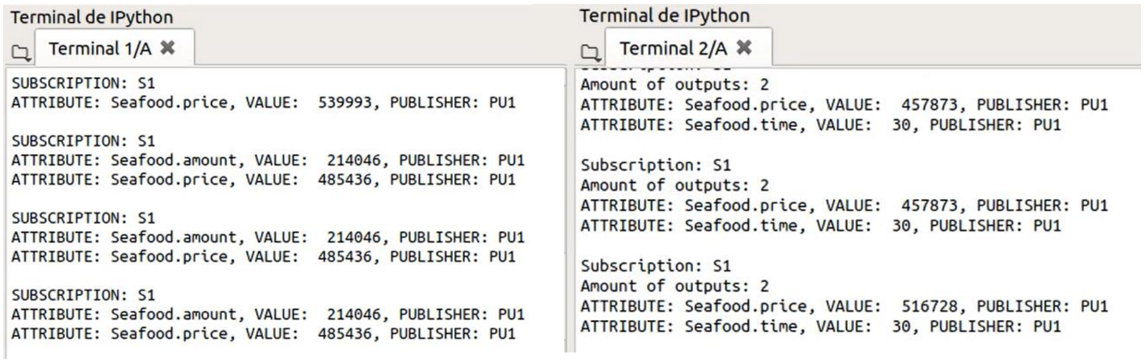


```
def SEAFOOD_ARRIVES_TO_WAREHOUSE():
 global RANDOM_VALUE
 global OCURRENCE
 global RFID_STATE
 global TIMESTAMP

 #We verify simulation bounds
 if(DAY <= SIMULATION_TIME and
 TIMESTAMP ==  "Next"):
     #We generate new random number
     RANDOM_VALUE = random.random()

 if(RANDOM_VALUE >= THRESHOLD):
     OCURRENCE = "Active"
     TIMESTAMP = "Stop"
     RFID_STATE = "On"
     RANDOM_VALUE = 0
     #Seafood insertion
```

Figure 5-7 Event to Python code (Noreña *et al.*, in process)



```
SUBSCRIPTION: S1
ATTRIBUTE: Seafood.price, VALUE:  539993, PUBLISHER: PU1

SUBSCRIPTION: S1
ATTRIBUTE: Seafood.amount, VALUE:  214046, PUBLISHER: PU1
ATTRIBUTE: Seafood.price, VALUE:  485436, PUBLISHER: PU1

SUBSCRIPTION: S1
ATTRIBUTE: Seafood.amount, VALUE:  214046, PUBLISHER: PU1
ATTRIBUTE: Seafood.price, VALUE:  485436, PUBLISHER: PU1

SUBSCRIPTION: S1
ATTRIBUTE: Seafood.amount, VALUE:  214046, PUBLISHER: PU1
ATTRIBUTE: Seafood.price, VALUE:  485436, PUBLISHER: PU1
```

```
Amount of outputs: 2
ATTRIBUTE: Seafood.price, VALUE:  457873, PUBLISHER: PU1
ATTRIBUTE: Seafood.time, VALUE:  30, PUBLISHER: PU1

Subscription: S1
Amount of outputs: 2
ATTRIBUTE: Seafood.price, VALUE:  457873, PUBLISHER: PU1
ATTRIBUTE: Seafood.time, VALUE:  30, PUBLISHER: PU1

Subscription: S1
Amount of outputs: 2
ATTRIBUTE: Seafood.price, VALUE:  516728, PUBLISHER: PU1
ATTRIBUTE: Seafood.time, VALUE:  30, PUBLISHER: PU1
```

Figure 5-8 Simulation in Python. (Noreña *et al.*, in process)

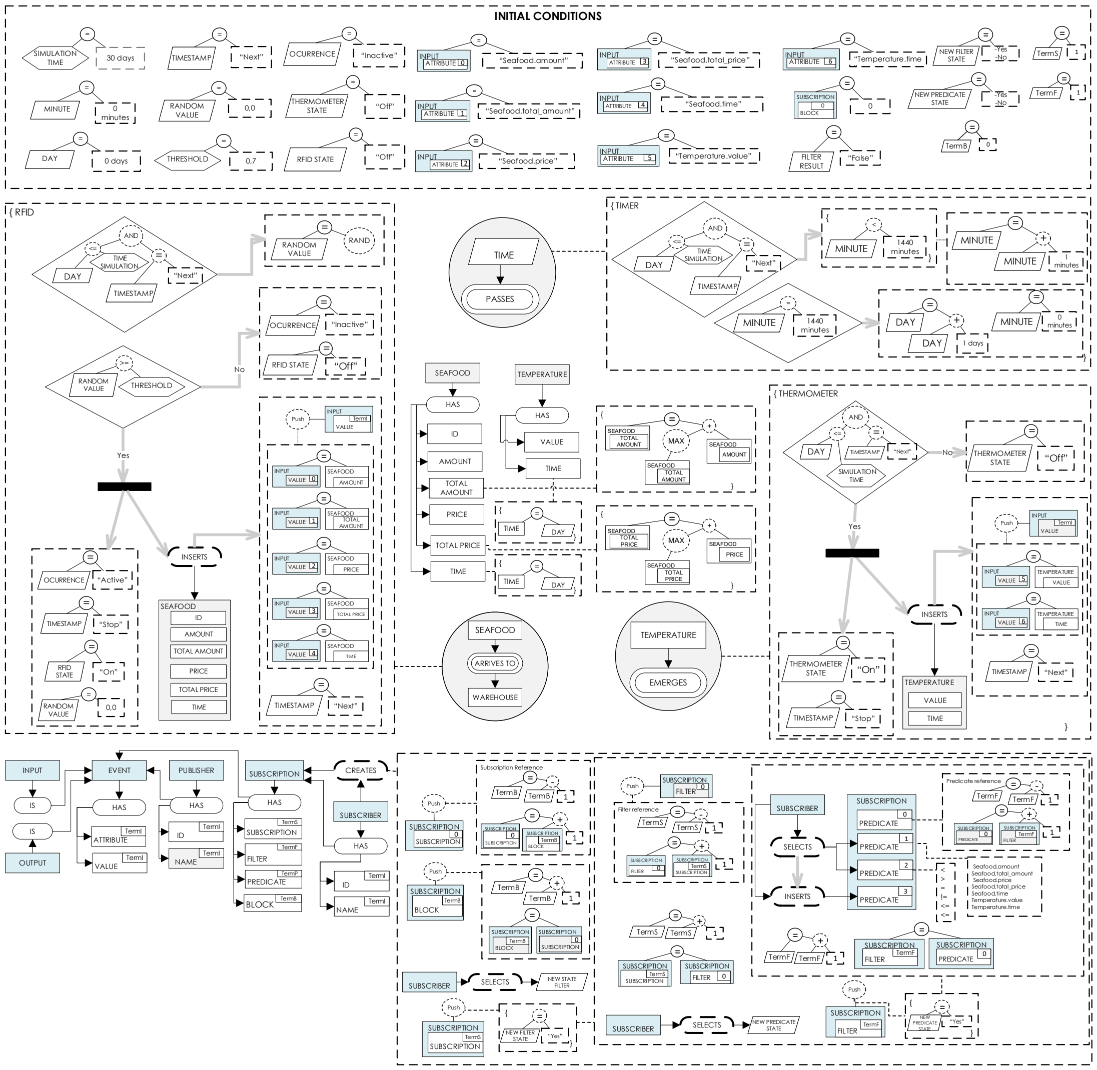


Figure 5-9 PCS applied to CEP (Noreña *et al.*, in process)

PL/SQL Code

PCS used in the lab study (in chemical domain, see Figure **4-33**) is translated into PL/SQL code (see event: *concentración de sustancia incrementa* in Figure **5-10** and Figure **4-32** in an English version). Such a translation is performed by Zapata-Tamayo (2019).

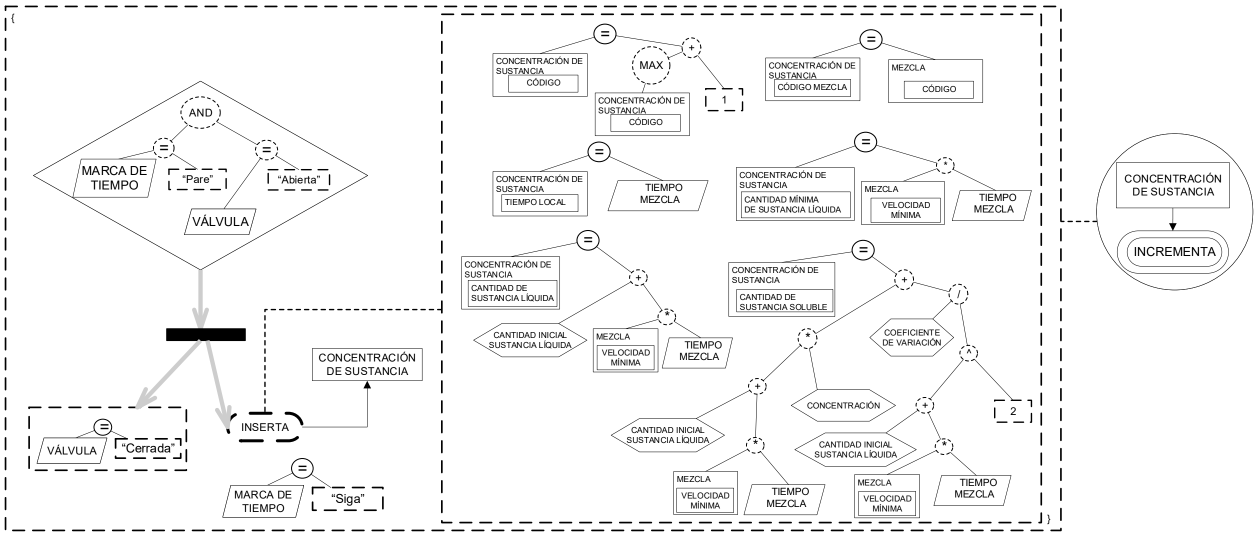

AND
MARCA DE TIEMPO
"Pare"
"Abierta"
VÁLVULA
VÁLVULA
"Cerrada"
INSERTA
CONCENTRACIÓN DE SUSTANCIA
MARCA DE TIEMPO
"Siga"
CONCENTRACIÓN DE SUSTANCIA
CÓDIGO
MAX
CONCENTRACIÓN DE SUSTANCIA
CÓDIGO
1
CONCENTRACIÓN DE SUSTANCIA
CÓDIGO MEZCLA
MEZCLA
CÓDIGO
CONCENTRACIÓN DE SUSTANCIA
TIEMPO LOCAL
TIEMPO MEZCLA
CONCENTRACIÓN DE SUSTANCIA
CANTIDAD MÍNIMA DE SUSTANCIA LÍQUIDA
MEZCLA
VELOCIDAD MÍNIMA
TIEMPO MEZCLA
CONCENTRACIÓN DE SUSTANCIA
CANTIDAD DE SUSTANCIA LÍQUIDA
CANTIDAD INICIAL SUSTANCIA LÍQUIDA
MEZCLA
VELOCIDAD MÍNIMA
TIEMPO MEZCLA
CONCENTRACIÓN DE SUSTANCIA
CANTIDAD DE SUSTANCIA SOLUBLE
COEFICIENTE DE VARIACIÓN
CONCENTRACIÓN
CANTIDAD INICIAL SUSTANCIA LÍQUIDA
CANTIDAD INICIAL SUSTANCIA LÍQUIDA
2
MEZCLA
VELOCIDAD MÍNIMA
TIEMPO MEZCLA
MEZCLA
VELOCIDAD MÍNIMA
TIEMPO MEZCLA
CONCENTRACIÓN DE SUSTANCIA
INCREMENTA

Figure 5-10 Event: *concentración de sustancia incrementa* (Noreña *et al.*, 2019)

```
CREATE TRIGGER "CONCENTRACION_DE_SUSTANCIA_I"
    AFTER UPDATE OF "MARCA_DE_TIEMPO" OR UPDATE OF "VALVULA" ON "VARIABLE"
DECLARE
    v_MARCA_DE_TIEMPO VARIABLE.MARCA_DE_TIEMPO%TYPE;
    v_VALVULA VARIABLE.VALVULA%TYPE;

    v_CODIGO_CONCENTRACION_DE CONCENTRACION_DE_SUSTANCIA.CODIGO%TYPE;
    v_CODIGO_MEZCLA_CONCENTRA
CONCENTRACION_DE_SUSTANCIA.CODIGO_MEZCLA%TYPE;
    v_CODIGO_MEZCLA MEZCLA.CODIGO%TYPE;


       v_CANTIDAD_MINIMA_SUSTANCIA_LI := v_VELOCIDAD_MINIMA_MEZCLA *
   v_TIEMPO_MEZCLA;

       v_CANTIDAD_DE_SUSTANCIA_S := ((V_CANTIDAD_INICIAL_SUSTANCIA_L +
   (v_VELOCIDAD_MINIMA_MEZCLA * v_TIEMPO_MEZCLA)) * v_CONCENTRACION) +
   (power((V_CANTIDAD_INICIAL_SUSTANCIA_L + (v_VELOCIDAD_MINIMA_MEZCLA *
   v_TIEMPO_MEZCLA)), 2) / v_COEFICIENTE_DE_VARIACIO);
```

```
    INSERT INTO CONCENTRACION_DE_SUSTANCIA (CODIGO,
CANTIDAD_DE_SUSTANCIA_LIQUIDA, CANTIDAD_MINIMA_DE_SUSTANCIA_L,
CANTIDAD_DE_SUSTANCIA_SOLUBLE, CODIGO_MEZCLA, TIEMPO_LOCAL)
    VALUES (v_CODIGO_CONCENTRACION_DE,
            v_CANTIDAD_DE_SUSTANCIA_LIQUID,
    v_CANTIDAD_MINIMA_SUSTANCIA_LI, v_CANTIDAD_DE_SUSTANCIA_S,
    v_CODIGO_MEZCLA,  v_TIEMPO_LOCAL_CONCENTRAC);

     END IF;
    END;

IF v_MARCA_DE_TIEMPO = 'PARE' AND v_VALVULA = 'ABIERTA' THEN
  v_CODIGO_CONCENTRACION_DE := v_CODIGO_CONCENTRACION_DE + 1;
  v_CODIGO_MEZCLA_CONCENTRA := v_CODIGO_MEZCLA;
  v_TIEMPO_LOCAL_CONCENTRAC := v_TIEMPO_MEZCLA;
  v_CANTIDAD_DE_SUSTANCIA_LIQUID :=  V_CANTIDAD_INICIAL_SUSTANCIA_L +
CANTIDAD_MINIMA_SUSTANCIA_LI;
```

Figure 5-11 Event to PL/SQL Code (Zapata-Tamayo, 2019)

C++ Code

A model based on the extended PCS is constructed in the petroleum engineering domain. Such a model includes complex mathematical notation and event representation, which were represented by using PCS (see some events in Figure **5-12**). The model is translated into

C++ code and simulated in a literature case (see code of the event: mesh appears in Figure **5-13**). Such a translation is performed by Velásquez (2019).

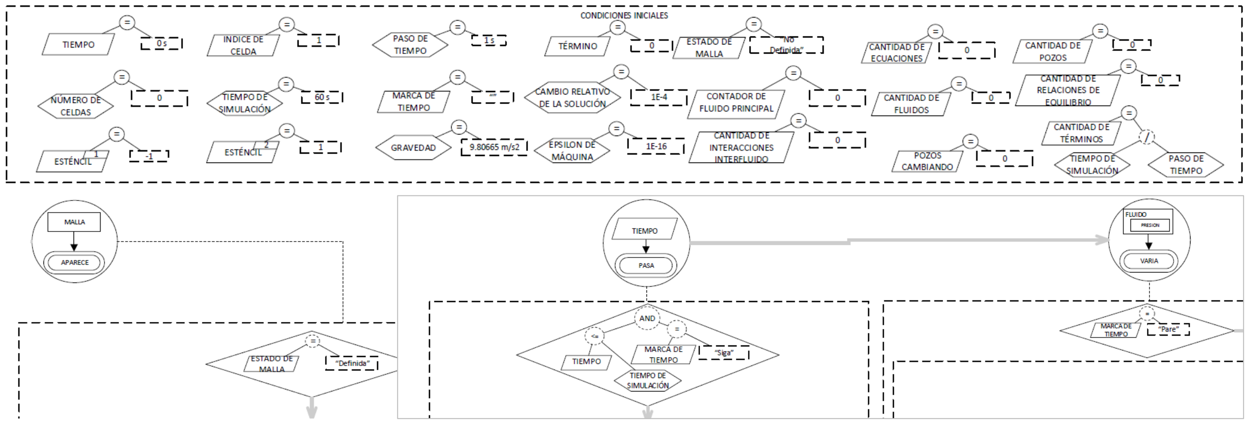
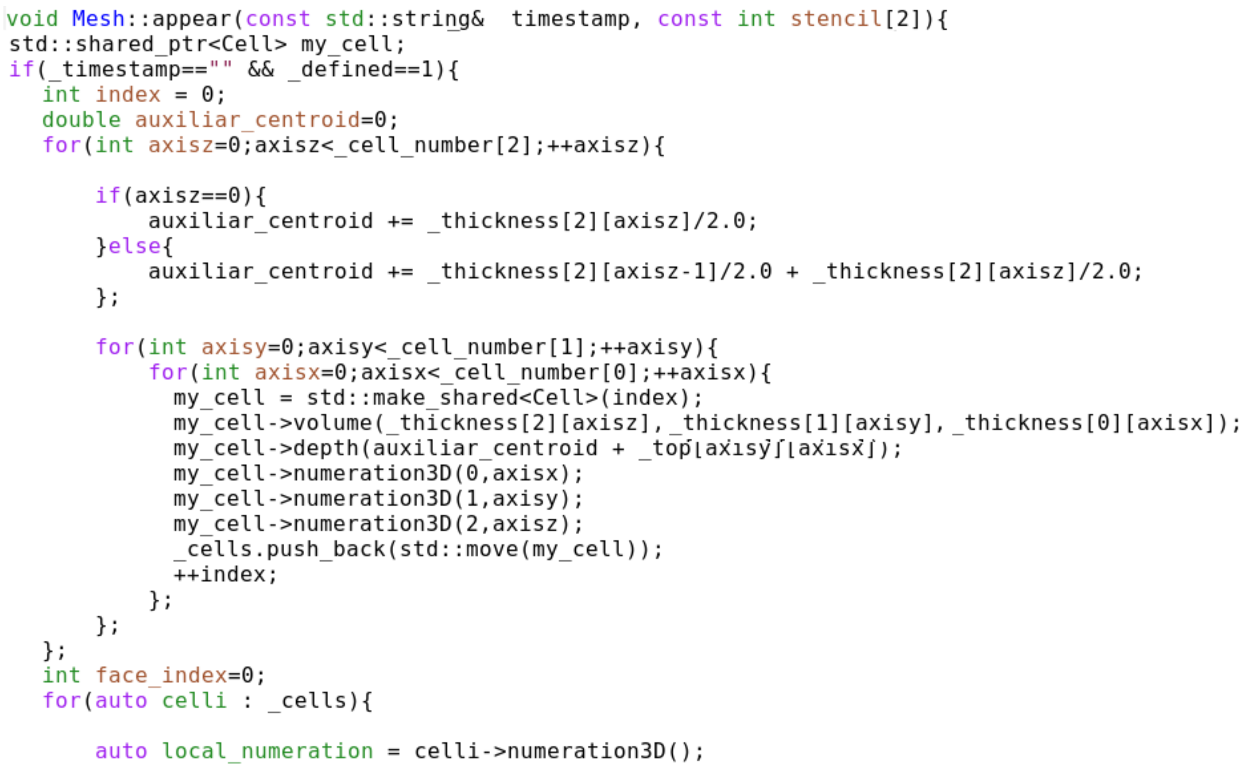


Figure 5-12 Event representation in petroleum engineering SSD (Velásquez, 2019)



```
void Mesh::appear(const std::string&  timestamp, const int stencil[2]){
std::shared_ptr<Cell> my_cell;
if(_timestamp=="" && _defined==1){
   int index = 0;
   double auxiliar_centroid=0;
   for(int axisz=0;axisz<_cell_number[2];++axisz){

       if(axisz==0){
           auxiliar_centroid += _thickness[2][axisz]/2.0;
       }else{
           auxiliar_centroid += _thickness[2][axisz-1]/2.0 + _thickness[2][axisz]/2.0;
       };

       for(int axisy=0;axisy<_cell_number[1];++axisy){
           for(int axisx=0;axisx<_cell_number[0];++axisx){
             my_cell = std::make_shared<Cell>(index);
             my_cell->volume(_thickness[2][axisz],_thickness[1][axisy],_thickness[0][axisx]);
             my_cell->depth(auxiliar_centroid + _top[axisy][axisx]);
             my_cell->numeration3D(0,axisx);
             my_cell->numeration3D(1,axisy);
             my_cell->numeration3D(2,axisz);
             _cells.push_back(std::move(my_cell));
             ++index;
           };
       };
   };
   int face_index=0;
   for(auto celli : _cells){

       auto local_numeration = celli->numeration3D();
```

Figure 5-13 C++ code by using PCS (Velásquez, 2019)

5.3. Publications

Research Projects

Representación de eventos en esquemas preconceptuales mediante roles semánticos y ecuaciones matemáticas. Tecnológico de Antioquia, Institución Universitaria and Universidad Nacional de Colombia (Hermes code 39365), 2017-2018.

Una extensión al esquema preconceptual para el refinamiento en la representación de eventos y la notación matemática. Universidad Nacional de Colombia (Hermes code 39886), 2017-2020.

Complex Event Processing by Using Pre-conceptual Schemas. University of Toronto, 2019.

Tutoring

Carolina Cárdenas & Daniel Bedoya. Mathematical Structures Integration in Software Engineering for Representing Events in Automatic Systems by using Pre-conceptual Schemas. Degree Project, Tecnológico de Antioquia. Institución Universitaria, 2018.

Research Internship

Middleware Systems Research Group, Department of Electrical and Computer Engineering, University of Toronto, Canada. Supervisor Arno-Hans Jacobsen. From January to April 2019.

Book Chapters

Noreña, P. A., Torres, D. M., & Zapata C. M. (2017). "Interoperabilidad dinámica entre sistemas basados en internet de las cosas: una representación a partir de esquemas preconceptuales", *Industria 4.0 Escenarios e impactos.* Medellín: Universidad de Medellín, 159–173.

Noreña, P. A. Zapata, C. M., & Villamizar, A. (2018). "Representación de eventos a partir de estructuras lingüísticas basadas en roles semánticos: una extensión al esquema preconceptual". Investigación e *Innovación en Ingeniería de Software 2*, Medellín: Publicar T, Sello editorial TdeA, 69–79.

Book

Zapata-Jaramillo, C. M., Noreña, P. A. & Zapata-Tamayo, S. *Especificación de las características estructurales, dinámicas, télicas y eventuales de los esquemas preconceptuales.* Medellín: Universidad Nacional de Colombia, in process.

Journal Papers

Noreña, P. A., Zapata, C. M., & Villamizar, A. (2019). Representing Chemical Events by using Mathematical Notation from Pre-conceptual Schemas. *IEEE Latin American Transactions*, 17(01), 46–53.

Noreña, P. A. & Zapata, C. M. (2019). Business Simulation by using Events from Pre-conceptual-Schemas. *Development in Business Simulation and Experiential Learning*, 46, 258-263.

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(6), 207–220.

Durango, C., 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(6), 327–341.

Noreña, P. A. & Zapata, C. M. (2018). A Game for Learning Event-Driven Architecture: Pre-conceptual-Schema-based Pedagogical Strategy. *Development in Business Simulation and Experiential Learning*, 45, 312–31.

Calle, J. Noreña, P. A., & Zapata, C. M. Extension to Pre-conceptual Schemas: A Set of New Mathematical Structures for Scientific Software Domain Representation. *Ingeniare*, in process.

Zapata-Jaramillo, C. M., Zapata-Tamayo, S., & Noreña, P. A. 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*, in process.

Noreña, P. A. & Zapata, C. M. Simulating Events in Requirements Engineering by using Pre-conceptual-Schema-based Components from Scientific Software Domain Representation. *IET Software*, in process.

Noreña, P. A., Scaunasu, B. Elliott, G., Jacobsen A. H., Zapata, C. M. & Mosquera, J. D. Complex Event Processing by Using Pre-conceptual Schemas. *IEEE Transactions on Services Computing*, in process.

Conferences

Noreña, P. A. & Zapata, C. M. (2017). Una extensión al esquema preconceptual para el refinamiento en la representación de eventos y la notación matemática. In *Latin American Software Engineering (LASES 2017)*, Medellín, Colombia. (Poster).

Noreña, P. A. Zapata, C. M., & Villamizar, A. (2018). Estructuras lingüísticas basadas en roles semánticos para la representación de eventos a partir del esquema preconceptual. In *Seminario Internacional de Investigación en Ingeniería de software (SEIIIS 2017)*, Medellín, Colombia.

Noreña, P. A. (2018). Una extensión al esquema preconceptual para el refinamiento en la representación de eventos y la notación matemática. In *XXI Ibero-American Conference on Software (CIbSE 2018)*, Bogotá, Colombia, 589–596. (Conference paper).

Noreña, P. A. & Zapata. (2018). A Pre-conceptual-Schema-based Representation of Time Events Coming from Scientific Software Domain. In *22nd World Multi-Conference on Systemics, Cybernetics and Informatics (WMSCI 2018)*, Orlando, US, 53–58. (Conference paper).

Noreña, P. A. (2018). Una extensión al esquema preconceptual para el refinamiento en la representación de eventos y la notación matemática. In *Tesis en Tres Minutos (3MT)*. Universidad Nacional de Colombia, Bogotá campus, Colombia.

Software Application

Noreña, P. A., Zapata, C. M., & Mosquera J. D. (2019). Software for CEP from PCS. Intellectual property registration: 13-75-118.

5.4. PCS Templates to Case Tools

PCS templates to case tools: *Draw.io*^{TM} and *Microsoft Visio*^{TM} are constructed as value added (see Figure **5-14**) for easy creating a PCS. The templates are files (on English and Spanish), which can be downloaded from the link `https://github.com/panorenac/PCS-Templates.git`.

PCS template to Draw.xml can be used by selecting the option *file* from the menu, selecting the option *open Library from* (selecting the location of downloaded file from the *device*/a website *i.e., Google drive, GitHub*, etc.), and selecting the button *open*. Then, the template should appear in the *shapes* (symbol place on the left side of the main screen; see Figure **5-14 a**).

PCS template to VISIO.vss can be used by selecting from the *shapes* (left side of the main screen), the option *more shapes*, selecting the option *open symbol gallery*, searching the downloaded file in the location, and selecting the button *open*. Then, the template should appear in the *shapes* (see Figure **5-14 b**).

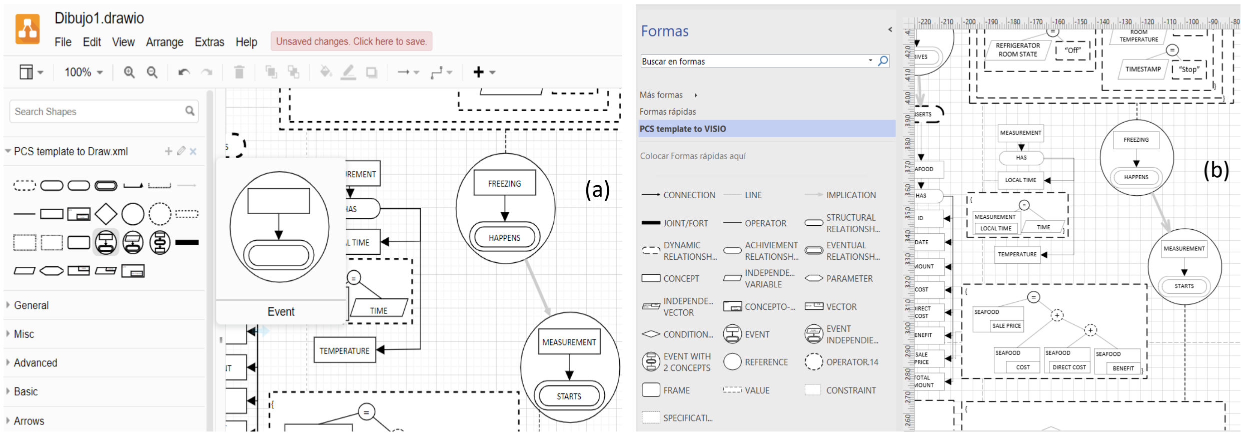


Figure 5-14 Templates to Draw.io **(a)** and Microsoft Visio **(b)**. The Authors

6 Conclusions and Challenges

Challenges are an opportunity to test you and rise to the next level.
—Angelica Montrose

6.1. Conclusions

The refinement of event representation and mathematical notation by using an extension to pre-conceptual schemas is proposed in this Ph.D. Thesis, which is achieved by obtaining the following contributions:

Related to characterizing events

Events were characterized by searching events emerging in SSD.

An *eventual verb list* (report) was constructed by using semantic roles (actants and circumstants) for events from computational linguistics and scientific modeling.

Related to defining structures

New scientific components are added to PCS notation.

New *linguistic structures* for PCS extension are based on the events characterized and 38 eventual verbs found.

New *graphical structures* for representing events (with zero to two actants) are defined according to the proposed linguistic structures.

New *mathematical structures* for PCS extension are extracted from mathematical models (complex equations) used in SSD. New *graphical structures* as: nodes (parameter, variable, vectors, and matrices), gatherers (initial condition and array table) and complex operators (mathematical, arrays, and trigonometrics) are defined for the mathematical structures by following the elements from the PCS notation.

Related to proposing an extension to PCS

An *extension to PCS* is proposed for refining event representation and mathematical notation in SSD.

Several SSD were represented in PCS by involving expert guidance.

PCS extension integrates scientific components; allows for representing the time and functionality of the events and structural and dynamic view of the elements of any SSD; understanding and recognition of the processes, events, and mathematical models in a SSD.

Business analysts, scientists, and students can use the PCS as computing models for representing SSD and their elements in software development and simulation.

Both software engineering and science fields are integrated in this Ph.D. Thesis. The extended PCS allows for integrating scientific and software components and reducing the gap between both fields.

Related to validating the extended PCS

An *experiment* with 36 scientist experts was carried out from universities and companies at Colombia in chemical domain for evaluating the understandability and usability of the proposed solution.

An *experiment* with 39 computing and simulation experts was carried out from universities and companies at Colombia, US, Mexico, Argentina, España, Paraguay, and China. Geology, electrical, and industrial domains were used for evaluating the understandability and usability of the proposed solution.

A *software application* for CEP from PCS is developed for evaluating the understandability and completeness of the proposed solution in the internship.

Validation results according to the statistical analysis were obtained from the consensus criteria by indicating the experts recognize and understand the extended PCS, its elements, and notation.

Experts also consider events, and mathematical notation are understandable. Experts also indicate the PCS can be used for representing SSD.

Research projects (3), tutoring (1), research internship (1), book chapters (2), journal papers (9), conferences (5), and software application (1) were performed as result publications of this Ph.D. Thesis.

We relate such conclusions with the methodology phases (solution and validation) and their activities and work products in Table **6-1**.

Table 6-1 Conclusions. The Authors

Methodology/ objectives	Activities	Work product: Finding	Conclusions: Contributions
CHARACTERIZING	Characterizing events	Event report Eventual Verb List	✓ Events were characterized by searching events emerging in SSD. ✓ An eventual verb list was constructed by using semantic roles (actants and circumstants) for events from computational linguistics and scientific modeling.
DEFINING	Defining linguistic structures	Linguistic structures	✓ New scientific components are added to PCS notation. New linguistic structures for PCS extension: 38 eventual verbs found.
Solution <phase>	Defining mathematical structures	Mathematical structures	✓ New mathematical structures for PCS extension are extracted from mathematical models (complex equations) used in SSD.
	Defining graphical structures	Graphical structures	✓ New graphical structures for representing events (with 0 to 2 actants) are defined according to the proposed linguistic structures. ✓ New graphical structures as nodes (parameters, variables, vectors, and matrices), gatherer (initial conditions), and complex operators (mathematical, array, and trigonometric) are defined by following the elements from the PCS notation.
PROPOSING	Including extension to PCS	Extension to PCS	✓ A PCS extension is proposed for refining event representation and mathematical notation in SSD. ✓ Several SSD were represented in PCS by involving expert guidance. ✓ PCS extension integrates scientific components; allows for representing time and functionality of the events and a structural and dynamic view; understanding and recognition of the processes, events, concepts, and mathematical models in a SSD. ✓ Business analysts, scientists, and students can use the PCS as computing model for representing SSD and their elements in software development and simulation. ✓ Both software engineering and science fields are integrated in this Ph.D. Thesis. The extended PCS allows for integrating scientific and software components and reducing the gap between both fields.
VALIDATING	Executing experiment	Experiment Software Application	✓ An experiment with 36 scientist experts was carried out from universities and companies at Colombia in the chemical domain for evaluating the understandability and usability of the proposed solution. ✓ An experiment with 39 computing and simulation experts was carried out from universities and companies in Colombia, US, Mexico, Argentina, España, Paraguay, and China in geology, electrical, and industrial domains for evaluating the understandability and usability of the proposed solution. ✓ A software app for CEP from PCS is developed in the internship.
Validation <phase>	Analyzing experiment	Experimental reports	✓ Validation results according to the statistical analysis were obtained from the consensus criteria by indicating the experts recognize and understand the extended PCS, its elements, and notation. ✓ Experts also consider events and mathematical notation are understandable. ✓ Finally, experts also indicate the PCS can be used for representing SSD.

6.2. Challenges

The following challenges are identified as future work from this Ph.D. Thesis:

PCS usability in industry, some students and analysts have presented and used PCS in software development from the industry. However, PCS are mostly used in academia. Analysts and scientists in industry can take advantage of the benefits of the PCS.

Requirements engineering of SSD, requirements engineering for events, needs from software engineering continue emerging. Therefore, best practices, methods, patterns, etc. can be assisted by modeling with PCS in the development process for SSD and event approaches, especially in requirements engineering, since this phase contains the problem solution. Such challenges would continue reducing the gap between science and software engineering fields.

Representation and simulation of other domains, other equations, several models, languages, and tools in modeling and simulation can be explored from PCS.

CEP applications modeling by using PCS, a first approach was performed in the internship, which can be improved according to data generation and event filtering techniques.

Event patterns, architectures Pub/Sub, architectures based on models, architectures based on events, and *distributed event-based systems* can be also represented by using PCS.

Events, conditions, and data can be modeled in *Databases* from PCS.

PCS exploration from *Neural networks and artificial Intelligence*, *e.g.*, machine learning and automated processes can be perfomed based on PCS.

Mathematical teaching by using PCS, a complete domain of a mathematical operation can be graphically represented in PCS. Then, PCS can be used as a tool in mathematical teaching processes.

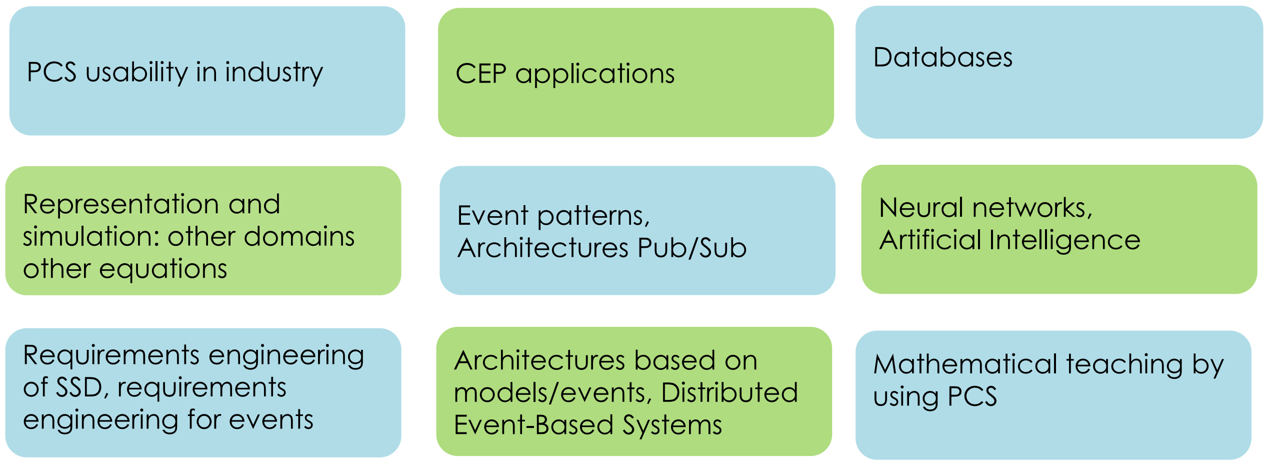

PCS usability in industry
CEP applications
Databases
Representation and simulation: other domains other equations
Event patterns, Architectures Pub/Sub
Neural networks, Artificial Intelligence
Requirements engineering of SSD, requirements engineering for events
Architectures based on models/events, Distributed Event-Based Systems
Mathematical teaching by using PCS

Figure 6-1 Challenges. The Authors

References

Amjad, A., Azam, F., Anwar, M. W., & Butt, W. H. (2017). Verification of Event-Driven Process Chain with Timed Automata and Time Petri Nets. In *9th IEEE-GCC Conference and Exhibition (GCCCE)*. IEEE. Manama, Bahrain, 1–6.

Armas-Cervantes, A., Baldan, P., Dumas, M., & Garcia-Bañuelos, L. (2016). Diagnosing Behavioral Differences between Business Process Models: An Approach based on Event Structures. *Information Systems*, 56:304–325.

Balkesen, C., Dindar, N., Wetter, M., & Tatbul, N. (2013). RIP: Run-based Itra-query Parallelism for Scalable Complex Event Processing. In *7th ACM International Conference on Distributed Event-based Systems*. ACM. Texas, US, 3–14.

Baouya, A., Bennouar, D., Mohamed, O. A., & Ouchani, S. (2015). A Probabilistic and Timed Verification Approach of SysML State Machine Diagram. In *12th International Symposium on Programming and Systems (ISPS)*. IEEE. Algiers, Algeria, 1–9.

Bazhenova, E., Zerbato, F., Oliboni, B., & Weske, M. (2019). From BPMN process models to DMN decision models. *Information Systems*, 83:69–88.

Bazydlo, G., Adamski, M., & Stefanowicz, Ł. (2014). Translation UML diagrams into Verilog. In *7th International Conference on Human System Interactions (HSI)*. IEEE. Costa da Caparica, Portugal, 267–271.

Beltrán-Saavedra, P. A. (2015). Precio del petróleo y el ajuste de las tasas de interés en las economías emergentes. *Borradores de Economía*, 901:1–37.

Boubeta-Puig, J., Díaz, G., Valero, V., & Ortiz, G. (2019). Medit4CEP-CPN: An Approach for Complex Event Processing Modeling by Prioritized Colored Petri Nets. *Information Systems*, 81:267–289.

Boubeta-Puig, J., Ortiz, G., & Medina-Bulo, I. (2015). Medit4CEP: A Model-Driven Solution for Real-time Decision Making in SOA 2.0. *Knowledge-Based Systems*, 89:97–112.

Burcea, S., Cică, R., & Bojariu, R. (2016). Hail Climatology and Trends in Romania: 1961–2014. *Monthly Weather Review*, 144(11):4289–4299.

Calle, J. M. (2016). Identificación de patrones de diseño para software científico a partir de esquemas preconceptuales. M.Sc. Thesis, Universidad Nacional de Colombia, Medellín Campus, Colombia.

Calle, J. M., Noreña, P. A., & Zapata, C. M. Extension to Pre-conceptual Schemas: A Set of New Mathematical Structures for Scientific Software Domain Representation. *Ingeniare*, in process.

Campos-Rebelo, R., Costa, A., & Gomes, L. (2015). Event Life Time in Detection of Sequences of Events. In *IEEE International Conference on Industrial Technology (ICIT).* IEEE. Seville, Spain, 3144–3149.

Chaverra, J. J. (2011). Generación automática de prototipos funcionales a partir de esquemas preconceptuales. M.Sc. Thesis, Universidad Nacional de Colombia, Medellín Campus, Colombia.

Chen, W., Wang, J., Shi, D., & Shi, L. (2017). Event based State Estimation of Hidden Markov Models through a Gilbert–Elliott Channel. *IEEE Transactions on Automatic Control*, 62(7):3626–3633.

Choi, E., Bahadori, M. T., Schuetz, A., Stewart, W. F., & Sun, J. (2016). Doctor ai: Predicting Clinical Events via Recurrent neural networks. In *Machine Learning for Healthcare Conference*. Durham, US, 301–318.

Chonoles, M. J. (2017). "Behavior: State Machine Diagrams." In *OCUP 2 Certification Guide* (313–342). ElSevier: Cambridge.

Da Silva, A. R. (2015). Model-Driven Engineering: A Survey Supported by the Unified Conceptual Model. *Computer Languages, Systems & Structures*, 43:139–155.

Dávid, I., Ráth, I., & Varró, D. (2018). Foundations for Streaming Model Transformations by Complex Event Processing. *Software & Systems Modeling*, 17(1):135–162.

Dayeh, M., Evans, N., Fuselier, S., Trevino, J., Ramaekers, J., Dwyer, J., Lucia, R., Rassoul, H., Kotovsky, D., Jordan, D., *et al.* (2015). First Images of Thunder: Acoustic Imaging of Triggered Lightning. *Geophysical Research Letters*, 42(14):6051–6057.

Drăghici, T., Negreanu, L., Bratu, O. G., Tincu, R., Socea, B., Iancu, M. A., Stănescu, A. M. A., & Diaconu, C. (2018). Liver Abnormalities in Patients with Heart Failure. *Archives of the Balkan Medical Union*, 53(1):76–81.

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.

Etzion, O., Fournier, F., Skarbovsky, I., & von Halle, B. (2016). A Model Driven Approach for Event Processing Applications. In *10th ACM International Conference on Distributed and Event-based Systems*. ACM. Irvine, US, 81–92.

Etzion, O., Niblett, P., & Luckham, D. (2011). *Event Processing in Action*. Stanford: Manning Publications Co.

Fillmore, C. J. (1971). Some Problems for Case Grammar. *Working Papers in Linguistics*, 10:245–265.

Fillmore, C. J. (1977). *The Case for Case Reopened in Syntax and Semantics*. New York: Academic Press In.

Gao, F., Curry, E., & Bhiri, S. (2014). Complex Event Service Provision and Composition based on Event Pattern Matchmaking. In *8th ACM International Conference on Distributed Event-Based Systems*. ACM. New York, US, 71–82.

Garrudo, F. (1990). Enlace mediante casos entre inglés y español. *Revista española de lingüística aplicada*, 6:9–18.

Gilbert, J. K. (2004). Models and Modelling: Routes to more Authentic Science Education. *International Journal of Science and Mathematics Education*, 2(2):115–130.

Giraldo, F. D., España, S., Giraldo, W. J., Pastor, Ó., & Krogstie, J. (2019). A Method to Evaluate Quality of Modelling Languages based on the Zachman Reference Taxonomy. *Software Quality Journal*, 27:1239–1269.

Gomaa, H. (2011). *Software Modeling and Design: UML, Use Cases, Patterns, and Software Architectures*. Cambridge: Cambridge University Press.

Gruber, J. S. (1965). *Studies in Lexical Relations*. Ph.D. Thesis, Massachusetts Institute of Technology, Boston, US.

Haas, W. (1960). Linguistic Structures. *Word*, 16(2):251–276.

Haisjackl, C., Soffer, P., Lim, S. Y., & Weber, B. (2018). How do humans inspect BPMN models: An Exploratory Study. *Software & Systems Modeling*, 17(2):655–673.

He, Y. (2016). Ontology-based Vaccine and Drug Adverse Event Representation and Theory-guided Systematic Causal Network Analysis toward Integrative Pharmacovigilance Research. *Current Pharmacology Reports*, 2(3):113–128.

Heaton, D. & Carver, J. C. (2015). Claims about the Use of Software Engineering Practices in Science: A Systematic Literature Review. *Information and Software Technology*, 67:207–219.

Heiko, A. (2012). Consensus Measurement in Delphi Studies: Review and Implications for Future Quality Assurance. *Technological Forecasting and Social Change*, 79(8):1525–1536.

Herzberg, N., Meyer, A., & Weske, M. (2013). An Event Processing Platform for Business Process Management. In *17th IEEE International Enterprise Distributed Object Computing Conference*. IEEE. Vancouver, Canada, 107–116.

Holey, E. A., Feeley, J. L., Dixon, J., & Whittaker, V. J. (2007). An Exploration of the Use of Simple Statistics to Measure Consensus and Stability in Delphi Studies. *BMC medical research methodology*, 7(1):52.

Howison, J., Deelman, E., McLennan, M. J., Ferreira da Silva, R., & Herbsleb, J. D. (2015). Understanding the Scientific Software Ecosystem and its Impact: Current and Future Measures. *Research Evaluation*, 24(4):454–470.

Jaramillo, C. M. & Esteban, P. V. (2006). Enseñanza y aprendizaje de las estructuras matemáticas a partir del modelo de van hiele. *Revista Educación y pedagogía*, 18:109–118.

Johanson, A. & Hasselbring, W. (2018). Software Engineering for Computational Science: Past, Present, Future. *Computing in Science & Engineering*, 20(2):90–109.

Kanewala, U. & Bieman, J. M. (2014). Testing Scientific Software: A Systematic Literature Review. *Information and software technology*, 56(10):1219–1232.

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.

Koltsov, V. B., Sevryukova, E. A., Yakovenko, D. V., & Kondrutieva, O. V. (2018). Physical-Chemical Modelling of the Ingredients of Air in the System of Monitoring Modern Industrial City. In *IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*. IEEE. Moscow, Russia, 1902–1906.

Kuipers, J., Ueda, T., & Vermaseren, J. (2015). Code Optimization in Form. *Computer Physics Communications*, 189:1–19.

Kuznetsov, E. & Merzlikin, A. (2019). The Surface Wave on the Boundary between a Hyperbolic Magnetooptical Single-axis Metamaterial and an Isotropic Dielectric. *Journal of Communications Technology and Electronics*, 64(3):223–228.

Lee, S. & Hwang, I. (2015). Event-based State Estimation for Stochastic Hybrid Systems. *IET Control Theory & Applications*, 9(13):1973–1981.

Li, Y. (2015). DRUMS: Domain-specific Requirements Modeling for Scientists. Ph.D. Thesis, Technische Universität München, München, Germany.

Li, Y., Guzman, E., Tsiamoura, K., Schneider, F., & Bruegge, B. (2015). Automated Requirements Extraction for Scientific Software. *Procedia Computer Science*, 51:582–591.

Liu, F. & Zhao, G. (2016). Monitoring of Software Project Progress base on Automata Theory. In *2nd Workshop on Advanced Research and Technology in Industry Applications (WARTIA-16).* Atlantis Press. Dalian, China, 404–409

Liu, W., Tan, Y., Ding, N., Zhang, Y., & Liu, Z. (2016). An Ontology Pattern for Emergency Event Modeling. In *IEEE 14th International Conference on Dependable, Autonomic and Secure Computing (DASC/PiCom/DataCom/CyberSciTech).* IEEE. Auckland, New Zeland, 151–156.

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

Luckham, D. (2011). *Event Processing for Business: Organizing the Real-time Enterprise.* New Jersey: John Wiley & Sons.

Luo, J. & Zhou, M. (2016). Petri-Net Controller Synthesis for Partially Controllable and Observable Discrete Event Systems. *IEEE Transactions on Automatic Control*, 62(3):1301–1313.

Meng, M., Ping, W., Chao-Hsien, C, & Ling, L. (2014). Efficient Multipattern Event Processing over High-speed Train Data Streams. *IEEE Internet of Things Journal*, 2(4):295–309.

Merkens, J.-L., Reimann, L., Hinkel, J., & Vafeidis, A. T. (2016). Gridded Population Projections for the Coastal Zone under the Shared Socioeconomic Pathways. *Global and Planetary Change*, 145:57–66.

Mezerins, A. (2014). Experimental Studies of Analog Signal Digital Representing based on a High Performance Event Timer. In *14th Biennial Baltic Electronic Conference (BEC).* IEEE. Tallinn, Estonia, 169–172.

Molaei, S. M. & Keyvanpour, M. R. (2015). An Analytical Review for Event Prediction System on Time Series. In *2nd International Conference on Pattern Recognition and Image Analysis (IPRIA).* IEEE. Rasht, Iran, 1–6.

Moreda, P. (2008). *Los roles semánticos en la tecnología del lenguaje humano: anotación y aplicación.* Ph.D. Thesis, Universidad de Alicante, Alicante, España.

Nanthaamornphong, A. & Carver, J. C. (2017). Test-Driven Development in Scientific Software: A Survey. *Software Quality Journal*, 25(2):343–372.

Navas, J. (2017). *Modelos matemáticos discretos en la empresa.* España: Universidad de Jaén.

Noreña, P. A. (2014). Un mecanismo de consistencia en los eventos disparador y de resultado para los artefactos de UNC-Method. M.Sc. Thesis, Universidad Nacional de Colombia, Medellín Campus, Colombia.

Noreña, P. A. (2018). An Extension to Pre-conceptual Schemas for Refining Event Representation and Mathematical Notation. In *XXI Ibero-American Conference on Software: CIbSE 2018.* Bogotá, Colombia, (45)589–596.

Noreña, P. A., Torres, D. M., & Zapata, C. M. (2017). "Interoperabilidad dinámica entre sistemas basados en internet de las cosas: una representación a partir de esquemas preconceptuales". In *Industria 4.0 Escenario e impacto* (159–173). Medellín: Sello Editorial Universidad de Medellín.

Noreña, P. A. & Zapata, C. M. (2018a). 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.

Noreña, P. A. & Zapata, C. M. (2018b). A Game for Learning Event-driven Architecture: Pre-conceptual-schema-based Pedagogical Strategy. *Developments in Business Simulation and Experiential Learning*, 45:24–37.

Noreña, P. A. & Zapata, C. M. (2018c). A Pre-conceptual-schema-based Representation of Time Events Coming from Scientific Software Domain. In *22nd World Multi-Conference on Systemics, Cybernetics and Informatics: WMSCI 2018.* Orlando, US, 53–58.

Noreña, P. A., Zapata, C. M., & Villamizar, A. E. (2018). "Representación de eventos a partir de estructuras lingüísticas basadas en roles semánticos: una extensión al esquema preconceptual". In *Investigación e Innovación, v.2* (69–79). Medellín: Publicar T, Sello editorial Tecnológico de Antioquia.

Noreña, P. A. & Zapata, C. M. (2019). Business Simulation by using Events from Pre-conceptual Schemas. *Developments in Business Simulation and Experiential Learning*, 46:258–263.

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.

Noreña, P. A., Scaunasu, B., Elliott, G., Jacobsen, H. A., & Zapata, C. M. Complex Event Processing by Using Pre-conceptual Schemas. *IEEE Transactions on Services Computing*, in process.

Noreña, P. A. & Zapata, C. M. Simulating Events in Requirements Engineering by using Pre-conceptual-Schema-based Components from Scientific Software Domain Representation. *Advances in Engineering Software*, in process.

Obi, K., Ramsey, M., Hinton, A., Stanich, P., Gray II, D. M., Krishna, S. G., El-Dika, S., & Hussan, H. (2018). Insights into Insulin Resistance, Lifestyle, and Anthropometric Measures of Patients with Prior Colorectal Cancer compared to Controls: A National Health and Nutrition Examination Survey (nhanes) Study. *Current Problems in Cancer*, 42(2):276–285.

OMG. (2011). *Superestructure 2.4.1.* OMG, Object Management Group. `http://www.omg.org/spec/UML/2.4.1`.

OMG. (2014a). *Business Process Model and Notation 2.0.* OMG, Object Management Group. `https://www.omg.org/spec/BPMN/About-BPMN/`.

OMG. (2014b). *Model Driven Architecture (MDA) Guide rev. 2.0.* OMG, Object Management Group. `https://www.omg.org/cgi-bin/doc?ormsc/14-06-01`.

OMG. (2015). *Superstructure 2.5.* OMG, Object Management Group. `http://www.omg.org/spec/UML/2.5/`.

Paddock, M. & Chapin, J. (2016). Bleeding diatheses: Approach to the Patient who Bleeds or has Abnormal Coagulation. *Primary Care: Clinics in Office Practice*, 43(4):637–650.

Patri, O. P., Sorathia, V. S., Panangadan, A. V., & Prasanna, V. (2014). The Process-oriented Event Model (PoEM): A Conceptual Model for Industrial Events. In *8th ACM International Conference on Distributed Event-Based Systems*. ACM. Mumbai, India, 154–165.

Payne, T. E. (1997). *Describing Morphosyntax: A Guide for Field Linguists.* Cambridge: Cambridge University Press.

Pozo, J. I. (2006). *Teorías cognitivas del aprendizaje.* Madrid: Ediciones Morata.

Ravikumar, G., Khaparde, S. A., & Joshi, R. K. (2016). Integration of Process Model and CIM to represent Events and Chronology in Power System Processes. *IEEE Systems Journal*, 12(1):149–160.

Reinartz, C., Metzger, A., & Pohl, K. (2015). Model-based Verification of Event-driven Business Processes. In *9th ACM International Conference on Distributed Event-Based Systems*. ACM. Oslo, Norway, 1–9.

Sahoo, A., Xu, H., & Jagannathan, S. (2015). Adaptive Neural network-based Event-Triggered Control of Single-input Single-output Nonlinear Discrete-time Systems. *IEEE Transactions on Neural Networks and Learning Systems*, 27(1):151–164.

Sarno, R., Wibowo, W. A., Solichah, A., *et al.* (2015). Time based Discovery of Parallel Business Processes. In *International Conference on Computer, Control, Informatics and its Applications (IC3INA)*. IEEE. Bandung, Indonesia, 28–33.

Shekarpour, S., Alshargi, F., Thirunaravan, K., Shalin, V. L., & Sheth, A. (2019). CEVO: Comprehensive EVent Ontology Enhancing Cognitive Annotation on Relations. In *IEEE 13th International Conference on Semantic Computing (ICSC)*. IEEE. Newport Beach, US, 385–391.

Sudars, K., Bilinskis, I., Boole, E., & Vedin, V. (2015). Signal Analog-to-event-to-digital Converting based on Periodic Sampling and Precise Event Timing. In *25th International Conference Radioelektronika*. IEEE. Pardubice, Czech Republic, 133–136.

Taiwe, G., Moto, F., Pale, S., Kandeda, A., Dawe, A., Kouemou, N., Ayissi, E., Ngoupaye, G., Njapdounke, J., Nkantchoua, G., *et al.* (2016). Extracts of Feretia Apodanthera del. demonstrated Anticonvulsant Activities against Seizures induced by Chemicals and Maximal electroshock. *Epilepsy Research*, 127:30–39.

Tarun, M., Kumar, V., Kumar, S., Jajoo, M. U., Rahman, S. U., & Sengupta, J. (2017). GPS and GSM based Rail Signaling and Tracking System. In *4th International Conference on Control, Decision and Information Technologies (CoDIT)*. IEEE. Barcelona, Spain, 0500–0504.

Tesnière, L. (1965). *Éléments de Syntaxe Structurale*. Paris: Klincksieck.

Treur, J. (2016). Dynamic Modeling based on a Temporal–causal Network Modeling Approach. *Biologically Inspired Cognitive Architectures*, 16:131–168.

Vásquez, A. S. & Sandova, E. L. (2017). Una comparación cualitativa de la dinámica de sistemas, la simulación de eventos discretos y la simulación basada en agentes. *Ingeniería Industrial*, 35:27–52.

Velásquez, S. (2019). 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.

Vose, R., Easterling, D. R., Kunkel, K., & Wehner, M. (2017). Temperature changes in the United States. Climate Science Special Report: A Sustained Assessment Activity of the U.S. Global Change Research Program, 267–300.

Wang, H., Lu, S., Zhang, C., Wang, Q., & Xu, F. (2016). Timing-IdeaGraph: A directed Cognition Graph Approach for Decision Making based on Temporal Event Sequences. In *IEEE 20th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. IEEE. Hsinchu, Taiwan, 322–326.

Wang, J., Chen, M., Shen, H., Park, J. H., & Wu, Z.-G. (2017). A Markov Jump Model Approach to Reliable Event-triggered retarded Dynamic output Feedback H∞ Control for Networked Systems. *Nonlinear Analysis: Hybrid Systems*, 26:137–150.

Wang, Y., Gao, H., & Chen, G. (2018). Predictive Complex Event Processing based on Evolving Bayesian Networks. *Pattern Recognition Letters*, 105:207–216.

White, R. & McCausland, W. (2016). Volcano-tectonic Earthquakes: A New Tool for Estimating Intrusive volumes and Forecasting Eruptions. *Journal of Volcanology and Geothermal Research*, 309:139–155.

Wieringa, R. J. (2014). *Design Science Methodology for Information Systems and Software Engineering*. New York: Springer.

Wiese, I. S., Polato, I., & Pinto, G. (2019). Naming the Pain in Developing Scientific Software. *IEEE Software*. DOI: 10.1109/MS.2019.2899838.

Wilson, G., Aruliah, D. A., Brown, C. T., Hong, N. P. C., Davis, M., Guy, R. T., Haddock, S. H., Huff, K. D., Mitchell, I. M., Plumbley, M. D., *et al.* (2014). Best Practices for Scientific Computing. *PLoS biology*, 12(1):e1001745.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2012). *Experimentation in Software Engineering*. New York: Springer.

Wonham, W., Cai, K., & Rudie, K. (2018). Supervisory Control of Discrete-Event Systems: A Brief History. *Annual Reviews in Control*, 45:250–256.

Wu, T. H., Pang, G. K.-H., & Kwong, E. W.-Y. (2014). Predicting Systolic Blood Pressure using Machine Learning. In *7th International Conference on Information and Automation for Sustainability*. IEEE. Colombo, Sri Lanka, 1–6.

Xia, H., Jiao, J., & Dong, J. (2019). Extend UML based Timeliness Modeling Approach for Complex System. In *International Conference on Mathematics, Modeling, Simulation and Statistics Application (MMSSA 2018)*. Atlantis Press. Shanghai, China, 1–6.

Xia, W., Junpeng, M., *et al.* (2014). Research on Flexible Business Process of Bank Modeling based on EPC. In *International Conference on Management of e-Commerce and e-Government*. IEEE. Shanghai, China, 54–60.

Xue, D. & El-Farra, N. H. (2016). Output Feedback-based event-triggered Control of distributed Processes with Communication Constraints. In *IEEE 55th Conference on Decision and Control (CDC)*. IEEE. Las Vegas, US, 4296–4301.

Xue, S., Wu, B., & Chen, J. (2013). LightEPC: A Formal Approach for Modeling personalized Lightweight Event-Driven Business Process. In *IEEE International Conference on Services Computing*. IEEE. Santa Clara, US, 1–8.

Zapata, C. M. (2007). *Definición de un esquema preconceptual para la obtención automática de esquemas conceptuales de UML*. Ph.D. Thesis, Universidad Nacional de Colombia, Medellín Campus, Colombia.

Zapata, C. M. (2012). *The UNC-Method Revisited: Elements of the New Approach*. Saarbrücken: Lambert Academic Publishing.

Zapata, C. M., Noreña, P. A., & Granados, N. E. (2013). Representación de eventos disparadores y de resultado en el grafo de interacción de eventos. *Ingenierías USBMed*, 4(2):23–32.

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*, 41:256–262.

Zapata-Tamayo, J. S. (2019). Generación semiautomática de código PL/SQL a partir de representaciones de eventos basadas en esquemas preconceptuales. M.Sc. Thesis, Universidad Nacional de Colombia, Medellín Campus, Colombia.

Zapata-Tamayo, J. S. & Zapata-Jaramillo, C. M. (2018). Pre-conceptual schemas: Ten Years of Lessons Learned about Software Engineering Teaching. *Developments in Business Simulation and Experiential Learning*, 45:250–257.

Zapata-Jaramillo, C. M., Zapata-Tamayo, S., & Noreña, P. A. 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*, in process.

Zhang, K. & Zhang, L. (2016). Observability of Boolean Control Networks: A Unified Approach based on the Theories of Finite Automata and Formal Languages. *IEEE Transactions on Automatic Control*, 61(9):2733–2738.

Zhang, Y., Liu, W., Ding, N., Wang, X., & Tan, Y. (2015). An Event Ontology Description Framework based on SKOS. In *IEEE 12th International Conference on Ubiquitous Intelligence and Computing*. IEEE. Bali, Indonesia, 1774–1779.

Zhao, X.-J., Yang, Y.-Z., Zheng, Y.-J., Wang, S.-C., Gu, H.-M., Pan, Y., Wang, S.-J., Xu, H.-J., & Kong, L.-D. (2017). Magnesium isoglycyrrhizinate blocks Fructose-induced hepatic NF-κb/NLRP3 Inflammasome Activation and Lipid metabolism disorder. *European Journal of Pharmacology*, 809:141–150.

Zhong, X. & He, H. (2016). An Event-triggered ADP Control Approach for Continuous-time System with Unknown Internal States. *IEEE transactions on cybernetics*, 47(3):683–694.

Zhu, L. (2018). Ontology Pattern of Trajectory Anonymity for Query events. In *International Conference on Sensor Networks and Signal Processing (SNSP)*. IEEE. Xi‘an, China, 457–461.