March - April 2020
ISSN: 0193-4120 Page No. 4052 - 4059

Pattern Analysis of the ECA Rule for Complex Event Processing in IoT Environments

Kwan Hee Han[1], Young Joon Ko[*2]
[1]Professor, Department of Industrial & Systems Engineering, Gyeongsang National University, 501 Jinju-daero Jinju-si Gyeongsangnam-do, 52828, Republic of Korea
[*2]Professor, Lift Engineering Department, Korea Lift College, 120 Unjeong 1-gil Geochang-eup Geochang-gun Gyeongsangnam-do, 50141, Republic of Korea
hankh@gnu.ac.kr[1], yjoonko@klc.ac.kr[*2]

Article Info
Volume 83
Page Number: 4052 - 4059
Publication Issue:
March - April 2020

Abstract
Background/Objectives: Efficient event processing shortens the processing time of events, and enhances the quality and availability of information by providing real-time visibility to organizations. In an era of IoT (Internet of Things).

Methods/Statistical analysis: For handling reactive systems, a kind of ECA (Event-Condition-Action) languages are mainly used for event processing since they have a concrete and intuitive foundation. However, existing ECA rules reveal some limitations when it deals with the processing of complex events such as the tight coupling of event producers and consumers.

Findings: The objective of this paper is to propose an ECA rule pattern classification to remedy current limitations of the ECA rule for complex event processing (CEP) effectively and efficiently, develop a prototype system to show the usefulness of proposed ECA rule pattern.

Improvements/Applications: Existing ECA rule is further classified into 3 parts, and each part is patterned to meet the current requirements for handling complex events in real time enterprise. Finally, event processing rule is executed by assembling the relevant elements of the 3 parts of the proposed ECA rule pattern.

Keywords: *ECA rule, Real-time Enterprise, Complex event processing, Event-driven architecture, Internet of Things*

Article History
Article Received*: 24 July 2019*
Revised*: 12 September 2019*
Accepted*: 15 February 2020*
Publication*: 26 March 2020*

1. Introduction

In an era of Internet of Things (IoT), huge diverse amount of data is occurring in real-time, and most data generated relate to events that have occurred. Several big data methodologies were developed to deal with volatile human generated data and the enormous amount of data generated from Machine-to-Machine (M2M) interactions based on IoT platforms [1]. In such a real-time enterprise environment, various real-time data streams that are much more unstructured are created, often in the form of series of event occurrences. Moreover, the amount of available event data is rapidly expanding because of the decreasing costs and increasing speed of computers and internet.

To provide M2M-based services in real-time, most organizations have had to handle a growing volume of heterogeneous business events and transactions continuously. Efficient event processing shortens the processing time of events, and enhances the quality and availability of information by providing real-time visibility to organizations. Flexibility gained from event

March - April 2020
ISSN: 0193-4120 Page No. 4052 - 4059

processing ensures a proactive response, which results in appropriate problem-solving. Bruns et al. addressed that complex event processing can be used as the linchpin technology for intelligent M2M system, and provided an event-driven architecture that is adapted to the IoT environment [2].

An event is an occurrence within a particular system or domain; it is anything that happened or is considered as having happened in certain environment [3]. The area of event processing is emerging recently driven primarily by the greater need of business enterprises to react quickly for better visibility into what is happening in their organization [4]. In particular, complex event processing (CEP) is a set of methods and tools for reacting to complex events, which are events that could only happen if lots of other events happen [5]. CEP also effectively supports the reducing latency for an individual business activity, as it enables to extract useful events from raw data streams in the business transactions [6]. In other words, a complex event is an event that combines, represents or denotes a set of other events that should be handled to respond effectively. Events in the real world occur simultaneously from a variety of sources and various events are considered to be a single event by combining events. Therefore, the development of an efficient CEP system is urgently needed in IoT environments. Event processing programming language is categorized into 2 styles as follows: (1) stream-oriented programming, and, (2) rule-oriented programming. The stream-oriented programming is based on the data flow programming, in which the vertices are processing elements, and the arcs represent data flowing between these vertices. Rule-oriented languages are further divided into sub-types: active rules, production rules, and logic programming [3]. Active rules, which are also known as event-condition-action (ECA) rules, are descended from the discipline of active databases. ECA rules are executed as follows: when an event occurs, evaluate conditions and, if they are satisfied, trigger an action. Event processing systems based on reactive rules and especially ECA rules, which execute actions as a response to the sensing of events, were extensively studied during the 1990s [7]. ECA languages are a concrete and robust paradigm for handling reactive systems. Essential features of an ECA language are reactive and reasoning capabilities, the possibility to express complex actions and events, and a declarative semantics.

In spite of the above-mentioned advantages, there are some limitations with existing ECA rules for the handling of complex events: (1) ECA rules have been usually adopted in a conventional request-response interaction, which is the basis of most service-oriented architectures, and is synchronous in nature. In this paradigm, event producers and consumers are considered to be the same. However, event producers and consumers exist separately in the real world, and they are independent and interact asynchronously. Therefore, the decoupling of event producers and consumers is necessary when applying the ECA rule for CEP. (2) In IoT environments, there are multiple event producers, so there must be some distinctions among event sources in the event part of the ECA rule. Moreover, the method of event detection from the event producer must be specified explicitly. (3) In the condition part of the ECA rule, temporal constraints must be dealt with as well as general condition logic checks. Moreover, the kinds of event occurrences must be pre-defined for proper condition checking. (4) In the action part of the ECA rule, if multiple actions are needed, the processing sequence and execution method of multiple actions must be specified. (5) There must be some distinctions among event sinks in the action part of the ECA rule to deal with multiple event consumers.

The objective of this paper is to propose an ECA rule pattern classification to improve the existing

limitations of the ECA rule to deal with complex events in IoT environments in real-time enterprise environment, and to develop a prototype system for the efficient processing of huge complex events.

The rest of the paper is organized as follows: Section 2 reviews related works on event processing. Section 3 describes a proposed ECA rule pattern for CEP. Section 4 describes the prototype system developed in this study. Finally, the last section summarizes the results and suggests directions for future research.

2. RELATED WORKS

Event-driven architecture (EDA) is defined as a software architecture pattern promoting the production, detection, consumption of, and reaction to events. To date, several EDA architectures have been proposed to deal with complex events. Well-known EDAs are the works of Etzion and Niblett [3] and Moxey *et al.* [4]. Etzion and Niblett proposed an event-processing network comprised of event producer, event consumer, event channel and event processing agent. The architectural components used by Moxey *et al.* are event producer, event consumer, event emitter, event bus, and event handler.

As a commercial system, Esper is a well-known software package based on the stream-oriented programming style, which uses script language for event processing called EPL (Event Processing Language) similar to SQL [8]. On the other hand, IBM's WebSphere business events [9] and RuleCore's RuleCore CEP server [10] are commercial systems based on the rule-oriented language, especially active rules.

Isazadeh *et al.* proposed intelligent rule learning for ECA rules to maintain the efficiency of the system in dynamic environments, and also presented a method that uses a combination of multi flexible fuzzy tree algorithm and neural network [11]. Decker *et al.* developed a graphical language for modeling composite events in business processes, called BEMN (Business Event Modeling Notation), which resolves some requirements. These included event conjunction, disjunction and inhibition, as well as the cardinality of events whose graphical expression can be factored into flow-oriented process modeling and event rule modeling [12]. Bækgaard and Godskesen presented a specification language that can be used to specify real-time triggering conditions in terms of complex event patterns. The proposed specification language can be used to formulate complex, triggering conditions for active rules in terms of event patterns that involve sequences, alternations, iterations, and parallel compositions [13]. Paschke and Kozlenkov surveyed reaction rule approaches and rule-based event processing systems and languages of the past decades [14]. Boubeta-Puig *et al.* proposed both a graphical domain-specific modeling language (DSML) for facilitating CEP domain definitions by domain experts, and a graphical DSML for event pattern definition by non-technological users [15].

In the application area of event processing to BPM (Business Process Management), Rajsiri *et al.* proposed an BPMN-based integration method of the event-driven approach and business process modeling approach by developing a cloud-based event-driven business process editor and simulator [16]. Dunkel *et al.* presented a reference architecture for event-driven traffic management systems, which enables the analysis and processing of complex event streams in real-time [17]. For transforming the streaming model, framework of streaming model transformations was proposed by integrating complex event processing, incremental model query and reactive event-oriented transformation methods [18].

3. ECA RULE PATTERN FOR CEP

The ECA rule was intensively studied to support automated rule triggering in response to the

March - April 2020
ISSN: 0193-4120 Page No. 4052 - 4059

occurrence and state change of events in an area of an active database. The basic ECA structure is "on event if condition, do action".

To remedy the above-mentioned limitations in Section 1, the existing ECA rule is patterned in this paper. To do this, the existing ECA rule is separated into 3 parts as follows: (1) E (Event)-part: To remedy the first and second problems of the current limitations, the event source is categorized and the method of event detection from event source is specified in this part. (2) C (Condition)-part: To remedy the third problems of the current limitations, the kinds of event occurrence and event condition including temporal constraints are specified. (3) A (Action)-part): To remedy the fourth and fifth problems of the current limitations, the event destination is categorized, and multiplicities of action as well as action type are specified.

A. Specification of event source and event detection (E-part)

In the E-part, various event sources are classified by their attributes, and method of event detection from event sources is also specified according to the role of event processing system as depicted in the left part of Table 1. Event sources are classified to 4 types as database, e-mail, file and smart device according to the characteristics of data storage media. In the database type, new record insertion of database table is treated as new event occurrence. For example, in the CRM (Customer Relationship Management) system, when new customer is registered in the database, 'customer registration event' is occurred. In the e-mail type, the arrival of new mail in a certain account is treated as new event occurrence. For example, e-mail for maintenance request from a customer is a new event occurrence in the computer manufacturer. In the file type, new line insertion or data input in a cell of spreadsheet software is treated as new event occurrence. Examples of file type are notepad and MS Excel. In the smart device type, the generation of signal from a smart device is treated as new event occurrence.

According to the role of event processing system in the event detection, a detection method is further classified to either push or full type. In a push type, event source takes an active part to notify event occurrence to event processing system by API (Application programming Interface) or external procedure of DBMS. In a pull type, event processing system detects event occurrence from event source by periodic polling. Pull type is further classified to 2 sub-types: (1) occurrence count, and (2) occurrence time. Occurrence count method detects event occurrence by using the changes of occurrence number between current polling and last visit time to event source system. If there is a difference in occurrence number, it is considered that new event is occurred. Occurrence time method detects event occurrence by using event occurrence time stamp between current polling and last visit time to event source system. If there is a new time stamp in current visit time, this event is treated as new occurrence.

B. Specification of event occurrences and event condition check (C-part)

In the C-part, event occurrences are classified by multiplicity of event occurrences, and event condition checks are specified according to the existence of temporal constraints, as depicted in the middle part of Table 1. Event occurrence is further classified into single and multiple occurrence sub-types according to the number of events for condition checks.

In condition checks for action execution, the normal event condition check compares the attribute value of an event to a reference value. When there are temporal constraints for event processing, the timing condition checks are applied. The timing condition check compares the attribute value to a reference value during the

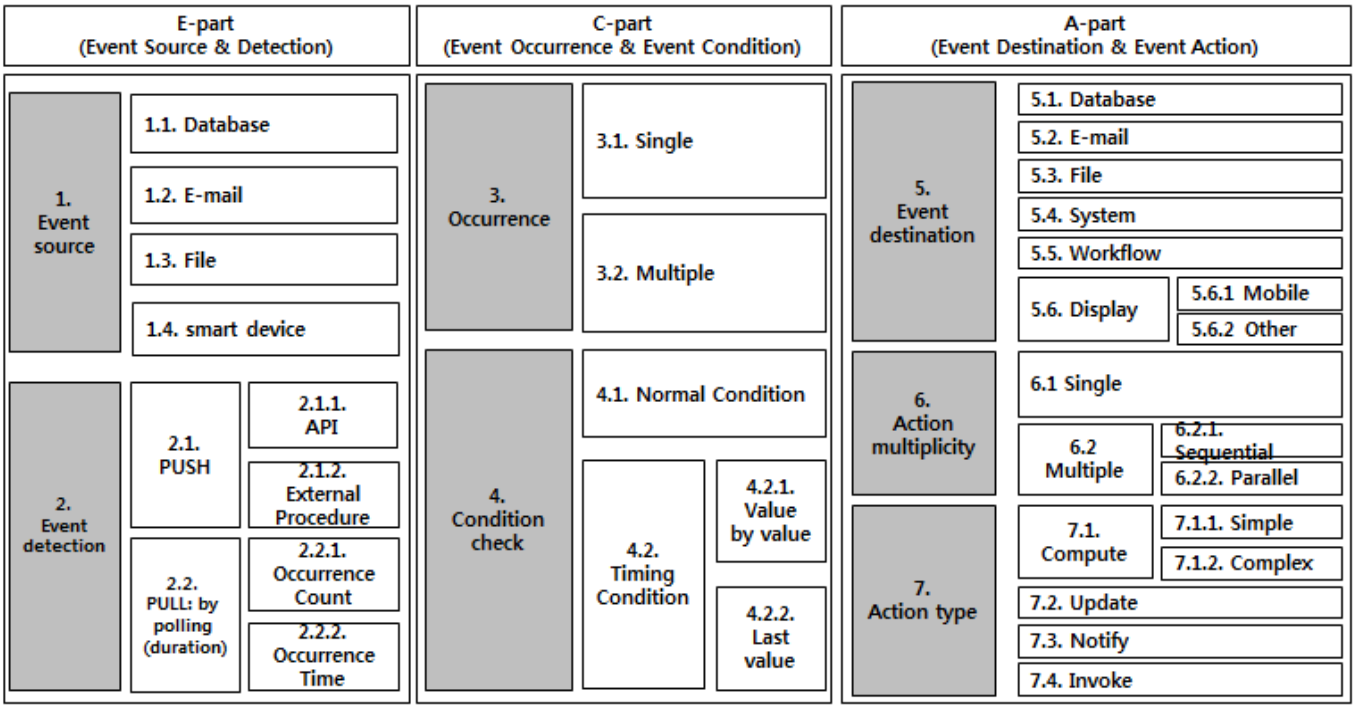
March - April 2020
ISSN: 0193-4120 Page No. 4052 - 4059

specified time period, and is further classified into 2 sub-types: (1) value by value comparison, and (2) last value comparison. In (1), whenever an event occurs within the pre-defined time periods, the attribute value of the new event occurrence is compared continually. For example, in a bridge management system (BMS) which monitors the bridge status continuously, if the wind velocity is greater than 50 m/sec just once during 5 minutes, the bridge has to be closed. In this case, value by value comparison during 5 minutes is used. In (2), after the specified time period, the last average value of the event is compared to a reference value only once. For example, in a BMS, if the average flow velocity is greater than 20m/sec at the end of every 5 minutes, the bridge has to be closed. In this case, the last value comparison after every 5 minutes is used.

C. Specification of event destinations and action types (A-part)

In the A-part, event destinations are classified by their attributes, and number of action needed action is specified. Moreover, the content of action is also classified as depicted in the right part of Table 1.

Table 1. Proposed ECA Rule Pattern



Event destination includes an application system, a workflow and a display as well as a database, a file and an e-mail that are previously defined in the event source specification. The display sub-type is further classified into mobile display and others. Number of action needed is classified into single and multiple action sub-types. Multiple actions are further classified into parallel or sequential action based on the required execution sequence among multiple actions. The content of action is classified into computation, update, notification and invocation.

Computation is further classified into the simple or complex sub-type. The simple sub-type is one of the four fundamental arithmetic operations. The complex sub-type is a combination of simple sub-types. Update action changes the existing content to a new one. Notification sends a message of process results to other systems or devices. Invocation sub-type runs other systems automatically as a result of the event processing.

March - April 2020
ISSN: 0193-4120 Page No. 4052 - 4059

4. DEVELOPMENT OF AN CEP SYSTEM BASED ON THE PROPOSED ECA RULE PATTERN

A prototype of the complex event processing system called CEpro was developed in this study based on the proposed ECA rule pattern. CEpro is responsible for event processing logic, which is defined by specifying each element of the 3 parts of the proposed ECA rule pattern, as explained in Section 3. Event producers, event consumers, and CEpro are independent from each other.

The CEpro consists of 4 modules: (1) the graphic modeler module specifies each element of the ECA rule pattern by graph structure. Nodes stand for event definition, event detection, event condition check, event action and start/stop of event processing. Graphic model is saved in XML format, which consists of 6 elements: Event-Define, Event-Detection, Event-Condition, Event-Action, Join-Split (node branch/merge information) and Flow (transition information). (2) the executor module processes event processing logic by interpreting XML format model file. (3) the scenario manager loads, saves, and retrieves

multiple event processing scenarios defined in the graphic modeler. (4) the monitor module shows the current status of event processing when the executor is running.

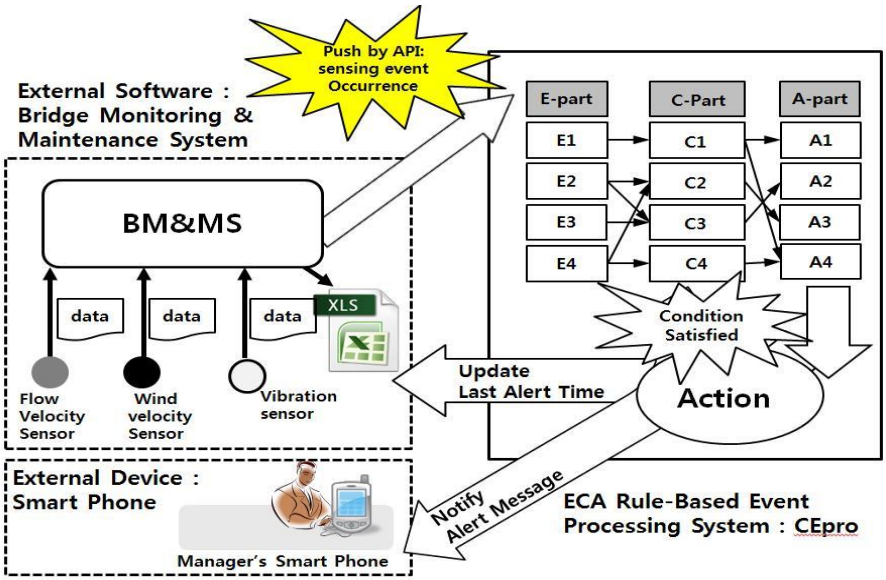


Figure 1. BM&M system and CEpro

Test scenario 1 is for a bridge monitoring and maintenance (BM&M) system, which monitors the bridge status in real-time, and takes proper actions for maintenance of the bridge. The relationship between the BM&M system and CEpro is shown in Figure 1.

For continual maintenance of a bridge, flow velocity beneath the bridge, wind velocity around the bridge, and vibration width of the bridge are sensed continuously, and recorded in specific cells in an Excel file.

If a sensor event is detected, it is transferred to CEpro by the API of the BM&M system. In CEpro, an event condition check is carried out as follows: For example, if {('flow-velocity' >= 30 after 120s) or ('wind velocity' >= 70 after 180s) or ('vibration_width' >=5 during 60s)}, then {(record the exception occurrence time to the Excel file within the BM&M system) and (notify the exception message to the manager by smart phone)}.

The ECA rule pattern of this scenario is summarized in Table 2.

March - April 2020
ISSN: 0193-4120 Page No. 4052 - 4059

Table 2. ECA rule pattern of test scenario 1

ECA part	element name	specifation	atribute
E-part	1. event source	smart device(1.4)	
	2. event detection	push(2.1)- API(2.1.1)	
C-part	3. event occurrence	multiple(3.2)	location, velocity, time stamp
	4. condition	normal condition (4.1)	if {(distance difference between registered location and current location>= 200m) or (duration of no signal transmission>= 5 minitues)}
A-part	5. event destination	display(5.6)-mobile(5.6.1)	
	6. action multiplicity	single (6.1)	
	7. action type	notify(7.3)	(1) display "warning: location break away" message to guardian's phone (2) display "warning:accident possible" message to guardian's phone

Test scenario 2 is a DGPS (Differential GPS)-based location monitoring and tracking (DLM&T) system. In this system, a DGPS module is installed in wheelchairs used by the elderly or infirm. The current location of the wheelchair is detected by the DGPS module and this location signal is transmitted to the smart phone of the guardian of the individual through the installed CEpro App.

Therefore, if the current location deviates from the registered area by more than 200m, a warning signal is displayed on the guardian's smart phone. In another situation, if the location signal is not displayed for 5 minutes, due to a possible accident, a warning signal is also transmitted to the guardian's smart phone.

The relationship between the DLM&T system and CEpro is shown in Figure 2, and the ECA rule pattern of this scenario is summarized in Table 3.

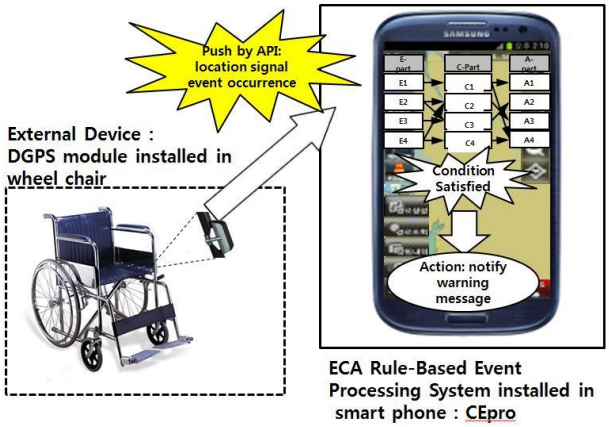


Figure 2. DLM&T system and CEPro

Table 3: ECA rule pattern of test scenario 2

ECA part	element name	specifation	atribute
E-part	1. event source	file(1.3)	
	2. event detection	push(2.1)- API(2.1.1)	
C-part	3. event occurrence	multiple(3.2)	flow velocity, wind velocity and vibration width
	4. condition	timing(4.2)-value by value (4,2,1)	if (average flow velocity sensing value>= 30 after 120s) or
			(average wind velocity sensing value >= 70 after 180s) or
		and last value(4.2.2)	(each vibration width sensing value >=5 during 60s)
A-part	5. event destination	file(5.3)	exception event occurrence time
		display(5.6)-mobile(5.6.1)	
	6. action multiplicity	multiple(6.2)-parallel(6.2.2)	
	7. action type	update(7.2)	update exception event occurrence time inExcel sensor input table
		notify(7.3)	display "exception occurred: bridge no.45" message to manager's phone

March - April 2020
ISSN: 0193-4120 Page No. 4052 - 4059

4. Conclusion

Currently, event processing systems have become one of the core elements for realizing real-time enterprise (RTE) through the monitoring and pattern recognition of incoming complex events. In this paper, by assembling subsets of ECA rule pattern, event processing rule is declared clearly to process heterogeneous complex events effectively and efficiently. A prototype CEP system was developed, and its operability was validated by using 2 test scenarios. In these scenarios, multiple events were detected, event conditions were checked, and the corresponding multiple actions were performed properly.

The ECA rule pattern in this paper is useful to practitioners and software developers who are implementing or developing a CEP system. However, improving and stabilizing the required functionalities for CEP, and a validity check of the proposed ECA rule pattern for complex event processing requires further research before implementing the proposed system in a real environment, since the developed system is just a prototype.

References

[1] I. Flouris, N. Giatrakos, A. Deliagiannakis, M. Garofalakis, M. Kamp, et al. M. Mock, Issues in complex event processing: status and prospects in the Big Data era,, Journal of Systems and Software, 2017; 127(1):217-36,.

[2] R. Bruns, J. Dunkel, H. Masbruch, S. Stipkovic, Intelligent M2M: complex event processing for machine-to-machine communication, Expert Systems with Applications, 2015;42(3):1235-46.

[3] O. Etzion, P. Niblett, Event processing in action, Stamford: Manning Publications Co; 2011.

[4] C. Moxey, H. Lalanne, G. Sharon, M. Peters, M. Edwards, O. Etzion, et al. M. Ibrahim, A conceptual model for event processing systems, IBM Red guide; 2010.

[5] D. Luckham, The power of events: An introduction to complex event processing in distributed enterprise systems, Berlin, Springer Verlag; 2008.

[6] I. Zappia, F. Paganelli, D. Parlanti, A lightweight and extensible Complex Event Processing system for sense and respond applications, Expert Systems with Applications, 2012; 39(12): 10408-19,

[7] A. Paschke, ECA-RuleML: An approach combining ECA Rules with temporal interval-based KR event/action logics and transactional update logics, IBIS, Technische Universität München, 2005. 30 p. Report No.: Technical Report 11/05.

[8] EsperTech[Internet], c2006, product overview technical data sheet; 2006[cited 2020 fab 07]. Available from: www.espertech.com/data-sheets/.

[9] J. Hoskins, Achieving business agility with IBM BPM and SOA connectivity, Gulf Breeze, FL, Maximum Press; 2010.

[10] M. Seiriö, M. Berndtsson, Design and implementation of an ECA rule markup language, Lecture Notes in Computer Science, 2005; 3791:98-112.

[11] A. Isazadeh, W. Pedrycz, F. Mahan, ECA rule learning in dynamic environments, Expert Systems with Applications, 2014; 41(17):7847-57.

[12] G. Decker, A. Grosskopf, A. Barros. A graphical notation for modeling complex events in business processes, 11th IEEE International Conference on Enterprise Distributed Object Computing Conference, 2007; 27-36, Annapolis, MD.

[13] L. Bækgaard, C. Godskesen, Real-time event control in active databases, Journal of Systems and Software, 1998; 42(3): 263-71.

[14] A. Paschke, A. Kozlenkov, Rule-based event processing and reaction rules, Lecture Notes in Computer Science, 2009; 5858: 53-66.

[15] J. Boubeta-Puig, G. Ortiz, L. Medina-Bulo, ModeL4CEP: Graphical domain-specific modeling languages for CEP domains and event patterns, Expert Systems with Applications, 2015; 42(21):8095-8110.

[16] V. Rajsiri, N. Fleury, G. Crosmarie, J. P. Event-based business process editor and simulator, Lecture Notes in Business Information Processing, 2011; 66: 707-18.

[17] J. Dunkel, A. Fernández, R. Ortiz, S. Ossowski, Event-driven architecture for decision support in traffic management systems, Expert Systems with Applications, 2011; 38(6): 6530-39.

[18] I. Dávid, I. Ráth, D. Varró, Foundations for streaming model transformations by complex event processing, Software & Systems Modeling, 2018; 17:135-62.