ELECTRONICS AND ELECTRICAL ENGINEERING
ISSN 1392 – 1215 2011. No. 7(113)
ELEKTRONIKA IR ELEKTROTECHNIKA

SYSTEM ENGINEERING, COMPUTER TECHNOLOGY
T 120
SISTEMŲ INŽINERIJA, KOMPIUTERINĖS TECHNOLOGIJOS

Ontology-Based Support for Complex Events

A. Sasa
Faculty of Computer and Information Science, University of Ljubljana,
Tržaška 25, Ljubljana, Slovenia, e-mail: ana.sasa@fri.uni-lj.si
O. Vasilecas
Information Systems Research Laboratory, Vilnius Gediminas Technical University,
Saulėtekio al. 11, 10223 Vilnius, Lithuania, e-mail: olegas.vasilecas@vgtu.lt

crossref http://dx.doi.org/10.5755/j01.eee.113.7.618

Introduction

With increasing demands for agility and reactivity of industrial and manufacturing processes, scientific circles and leading ICT (information and communications technology) companies have paid a lot of attention to EDA (event-driven architecture) and CEP (complex event processing). EDA and CEP enable event-driven systems - systems in which actions result from business events 3. Despite the recognized need for support of events to improve factory automation, signal processing, data acquisition, manufacturing, and other applications of industrial electronics, the use of EDA and CEP approaches in real-world solutions remains limited. One of the reasons for this is insufficient support for semantics in capturing and defining of complex events (CE) 20. Existing EDA and CEP approaches do not take into consideration different expressivity requirements that are needed in definition of a large number of diverse CE. We observe that semantic descriptions providing effective and expressive models for understanding of CE structures and their processing can be very helpful. In this paper, we show how achievements in the field of semantic technologies can contribute to the field of CEs and their support in information systems, especially from the point of view of their definition and processing. We present a framework that enables highly expressive event models and is based on ontologies and the Web Ontology language (OWL). We see ontologies and ontology representation languages as an opportunity to effectively deal with CEs in EDAs. Ontologies intrinsically provide a means for highly expressive semantic descriptions. In our framework, they are used to semantically describe CEs through conceptualizations of member events they are composed of. An important characteristic of our approach is that existing information that is captured in ontology can be reused in event detection and thus a wider scope of information can be taken into consideration when determining whether a CE should be triggered. Furthermore, ontologies enable hierarchical definitions of CEs and event taxonomies, which facilitates working with events on different levels of abstraction. If not all information for determining the exact subclass of event is available at a certain point of time, their generalizations can be used in order to increase the level of proactive behaviour of the overall system. When the information is added to the ontology their specializations are derived and used automatically. We use the OWL as the ontology representation language, because it provides a very appropriate foundation for CE definitions and has a wide support for reasoning. We propose a service that makes part of an EDA and enables translation of event data to OWL, detection of CEs and their triggering. It can act as an event source and as an event sink, which makes our approach complementary to existing approaches to support CEs that require higher expressivity and semantic descriptions. Our framework has been used in a case study project for electrical distribution domain, where it has been shown to be very useful and has improved the overall system flexibility and reactivity.

Background and related work

Events, event-driven architecture and complex event processing. Some of the biggest improvements in business processes and information systems in the current era are esteemed to be achieved through the discipline of event processing 4. An entity is considered event-driven when it acts in response to an event. EDA is a paradigm that describes an approach to information systems development with a focus on developing an architecture that has the ability to detect events and react intelligently to them 19. EDA represents a complement to the service-oriented architecture (SOA), which has become one of the most recognized paradigms in information systems development in the recent years 17. By enhancing the paradigm of SOA, enterprises can gain improve their ability for business transformation 21 by implementing event-driven architectures that automatically detect and react to significant business events 419. An important part of every EDA that enables and predetermines to what level a system is able to detect and respond to CEs is complex event processing (CEP). CEP is computing that performs

operations on CEs, including reading, creating, transforming, or abstracting them 10. CEP systems can be classified as advanced decision support systems 15. In comparison with other types of decision support systems, such as data mining based decision support systems that are not event driven 16, CEP systems focus on and on increasing system reactivity and proactivity based on information carried by member events. An event pattern is a template that matches certain sets of events 11. Events can be simple or complex 10:

- A simple event is as an event that is not an abstraction or composition of other events;
- A complex event is an event that is an abstraction of other events called its members.

Ontologies. An ontology represents an explicit specification of a conceptualization 7. There are several languages available for ontology representation, such as DAML, CGs, OIL, DAML+OIL, OWL (OWL 1, OWL 2). To support definition and processing of events, our work is based on OWL 2 extended with SWRL (Semantic Web Rule Language. The reason for this is that OWL represents a rich and useful group of ontology features for defining and describing relations and concepts 9, a high level of support, and its XML foundations, which make it appropriate to be used in conjunction with other Web technologies. The main concepts of OWL are Classes, Properties, and Individuals. This means that besides an the ontology that provides a conceptual representation, an OWL document can also comprise instance level descriptions of an enterprise. SWRL enables definition and processing of rules - implications between antecedents and consequents.

EDA and CEP approaches based on semantic technologies. The use of ontologies in the field of EDA and CEP has been identified as a suitable approach for the semantic definition of events by many researchers. Cheng et al. 5 have proposed a framework for context-aware processing of business rules in event-driven architectures. They have developed a context ontology in order to resolve a problem of inconsistent dictionaries at knowledge sharing and merging of rules. A key difference between their approach and ours is that they focus on achieving semantic interoperability, whereas our goal is to use ontology to define and detect CEs. Sen and Ma 18 have proposed an approach for combining reactive rules with ontologies. They have used ontologies to capture the context in which certain behaviour is appropriate, together with techniques for finding similarities with the primary objective to enable detection of similar CE patterns. Their research is related to our approach with the key difference that they do not talk about how CE types can be defined by the ontology, and how they can be used for detection in the context of EDA. Moser et al. 12 have proposed semantic correlation of events by using ontologies. Their work is based on the observation that the event correlation is necessary for CEP to relate events obtained from various sources in order to detect patterns and situations of the business context. Paschke 13 has proposed a language for semantic design of CEP patterns. The purpose of the language is to provide a description of successful CEP designs and to build libraries of best practice CEP descriptions and patterns. Adaikkalavan and Chakravarthy 1 propose an interval-based event specification language developed by expressing simple and composite events that are part of active rules. By contrast with these two approaches, we propose an EDA framework using an existing ontology language, which allows for usage of reasoners and other available tools that support the standard OWL language. Several authors propose definition and detection of events by using ontologies in the field of video multimedia data; examples are 2 and 6. On the other hand, our work is generic and not domain specific.

Architectural framework

In order to enable CE definitions, detection and triggering based on ontologies and the OWL language, we have developed an event-driven architectural framework. It defines a Complex Event Service (CES) that makes part of the overall event-driven service oriented architecture. The CES is strictly event driven: it is an event sink and an event source, and its interface does not define operations such as is the case with Web services. It catches events and triggers CEs when it detects them based on their definition and member events occurrences.

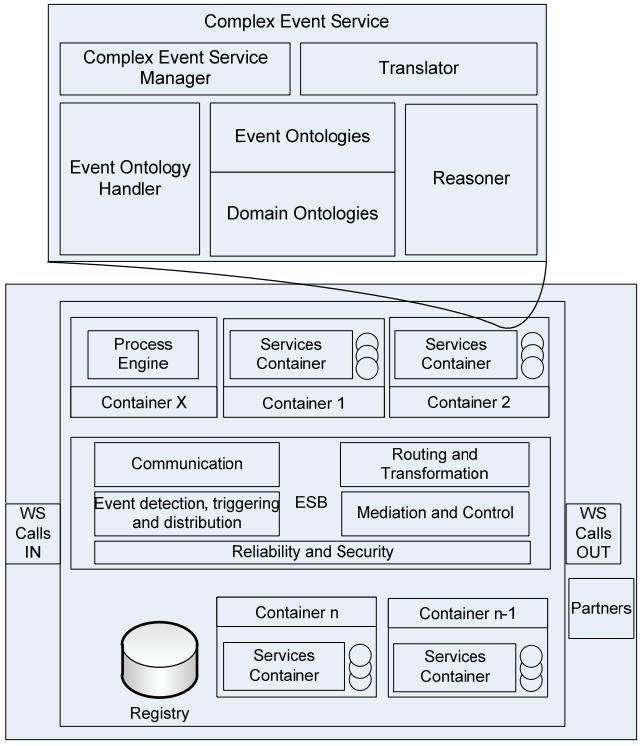

Fig. 1. Overall event-driven system architecture 8 and the Complex Event Service

Fig. 1 illustrates the structure of the CES and places it into the overall scope of EDA. The proposed architecture extends the architectural model of an enterprise service bus (ESB) based service-oriented system architecture proposed by Juric et al. in 8. ESB has become an important integration technology 22; in the scope of our work, it plays a central role in event detection, triggering and distribution. The CES is responsible for detection of CEs based on their complex event definitions and member event occurrences:

- The CES acts as an event sink for member events of the CEs it is responsible for;
- The CES acts as an event source for the detected CEs.

The CES consists of five components: Complex Event Service Manager, Translator, Ontology, Event ontology handler, and Reasoner. Ontology is a passive entity that is handled by the Event Ontology Handler and used by the Reasoner to infer new information based on the existing information in the ontology. It comprises information about the domain, about events and event types. An event ontology supports one CE type or several related CE types. We say that an event ontology is subscribed to all events that are members of the CEs that the ontology defines. Complex Event Service Manager is the component that links the CES with its environment by receiving member events and triggering detected CEs. It also orchestrates the other four active components of the CES into a complete process. The main process flow of the CES and the main responsibilities delegated to each component are illustrated in 0using the BPMN notation (Business Process Model and Notation). In the following sections, the four components orchestrated by the CES Manager are discussed in more detail. For better understanding, the discussion is supported by examples from our case study for the domain of electrical distribution companies. In the examples, we focus on events concerning electricity consumption predictions. In the case study, we have used Java EE and JMS to implement the CES. We have chosen to use Pellet as the reasoner, due to the most advanced support for both OWL and SWRL.

Ontology (event ontologies and domain ontologies)

Our overall ontology is composed of ontologies structured into two layers: event ontologies and domain ontologies. This paper concerns creation of event ontologies that enable definition of CEs and provide the basis for their detection. Domain ontologies comprise information about the domain where the system is used, such as the electrical distribution domain. For creation and maintenance activities of the domain ontologies, different generic ontology development methodologies that are available can be used. Example concepts of the electrical distribution domain ontology relevant for an increased electricity consumption prediction event are illustrated in Fig. 3.

The basic structure of an event ontology is defined by the base event ontology (Fig. 4). The base event ontology defines the class Event with four high level event subclasses: two disjoint subclasses that represent complex and atomic (simple) events (ComplexEvent and AtomicEvent), and two disjoint subclasses used to distinguish between instantaneous events and events that have a duration (non-instantaneous events). Basic temporal properties are defined upon the event classes. A non-instantaneous event can be defined either by start time and finish time, by start time and duration, or by finish time and duration. An object property that relates CEs to its member events is defined as follows

hasMemberEvent: ComplexEvent → Event

Every event ontology imports at least one domain ontology, the base event ontology and can import one or more other event ontologies. Event ontologies define subclasses of the ComplexEvent and AtomicEvent classes. Necessary and sufficient membership conditions have to be defined for every ComplexEvent. Domain ontology and event ontology concepts can be used to define these conditions.

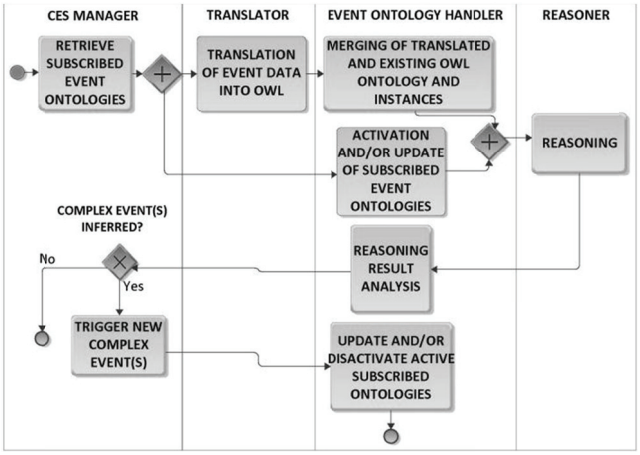

Fig. 2. Main process flow of the Complex Event Service

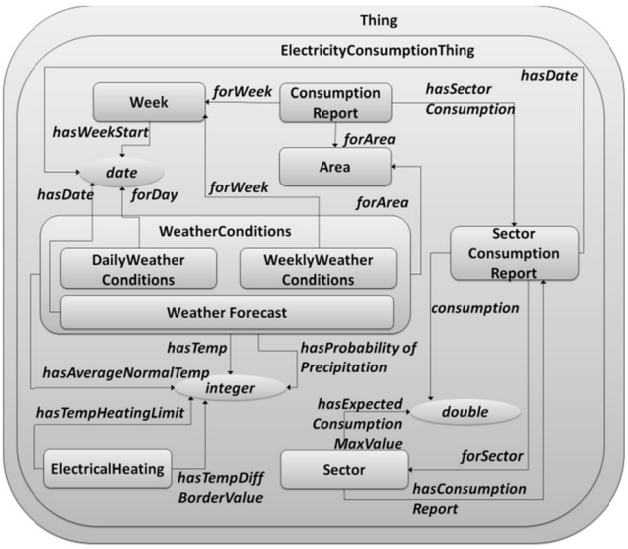

Fig. 2. Example concepts from electricity consumption domain ontology

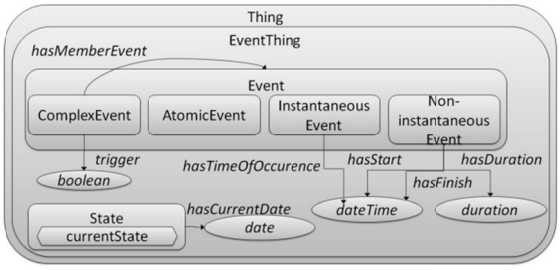

Fig. 3. Base event ontology

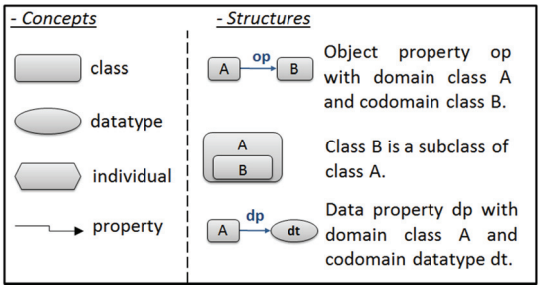

Fig. 4. Legend for figures 3 and 4

Thus, the necessary and sufficient membership conditions represent the join between the domain and the event concepts.

Domain ontology, member event types and their relevant properties may not always be enough to define a CE. CEs may not only depend on occurrences of other events, but also on the current state of some entity or a general state of the system (not directly related to any other concept in the ontology). For example, two events may occur that represent a CE in some state of the overall system, even though this may not be the case in some other state of the system. For such cases, the class State is defined with an individual *currentState*, representing the state of the overall system and its environment. An example property of this class is

currentDate: State → date.

The data property

trigger: ComplexEvent → boolean

is used to determine when a CE should be triggered. When a member of the ComplexEvent class has the positive value of this property, the Complex event service manager triggers the event.

In the electrical distribution domain, we demonstrate a semantic definition of an increased electricity consumption prediction event. An increased consumption prediction event is a CE that occurs when certain conditions apply that indicate that the consumption of electricity will be increased compared to planned or expected consumption. There are many conditions that can apply. In this paper, we take an example of weather conditions. Due to unexpected fall in temperature or snow before winter, electric heating is the only source of energy or the most used source of energy in certain areas, because charcoal and biomass are mostly saved for winter and have different terms of payment. Therefore, it is highly probable that the electricity consumption in the household sector will increase in such cases. An example SWRL statement indicating when an increased consumption prediction event for the household sector should be triggered is

IncreasedConsumptionPredictionEvent(*?e*) ∧ hasCurrentDate(*?s*, *?cd*) ∧ WeatherConditionsForecast(*?nwf*) ∧ hasDate(*?nwf*, *?cd*) ∧ ExtremelyColdWeather(*?nwf*) → trigger(*?e*, *true*) ∧ inSector(*?e*, *householdSector*).

Furthermore, if the electricity consumption has increased for this reason and the weather forecast indicates that the weather will worsen, one can predict that the consumption will remain increased or will increase further:

IncreasingConsumptionPredictionEvent(*?e*) ∧ hasCurrentDate(*?s*, *?cd*) ∧ DailyWeatherConditions(*?cwc*) ∧ forDay(*?cwc*,*?cd*) ∧ ExtremelyColdWeather(*?cwc*) ∧ WorseningWeatherConditionsForecast(*?nwf*) ∧ hasDate(*?nwf*, *?cd*) → trigger(*?e*, *true*) ∧ inSector(*?e*, *householdSector*).

The ontology concepts used in CE definitions can be defined recursively. For example, ExtremelyColdWeather can be defined as weather conditions where:

- The temperature difference between the average weather temperatures expected for the given time and the given temperatures is negative and less than the border value;
- The given temperatures is below the temperature limit indicating when the heating is required.

Example SWRL rule that defines this is:

ElectricalHeating(?eh) ∧ WeatherConditions(?wc) ∧ hasTemp(?wc, ?at) ∧ hasTempDiffBorderValue(?eh, ?bvt) ∧ hasAverageNormalTemp(?wc, ?nat) ∧ subtract(?diff, ?at, ?nat) ∧ lessThan(?diff, ?bvt) ∧ hasTempHeatingLimit(?eh, ?thl) ∧ lessThan(?at, ?thl) → ExtremelyColdWeather(?wc).

Translator

When a relevant member event is passed to the CES, the information that this event carries is translated to the OWL knowledge base (KB) containing OWL ontology and existing OWL instance level data. In the most basic form, event data carries information about the occurrence of the event and its temporal information, such as when it occurred or when it started or ended, if the event has a duration. However, in most cases the event data also carries other event-related information. Relevant event data may need to be merged with the existing data contained in OWL. To accomplish this, translations need to be specified between the event payload that carries the new data and between the OWL ontology concepts. Based on the translations, the data is translated into OWL. What part of data should be translated depends on the required information that has to be known in order to be able to determine if and when the CE occurs or does not occur. The translation of event data to OWL affects the instance level of the OWL document and not its ontology (conceptual level).

Table 1. Example of translations for the member event DailyWeatherForecastEvent

```
<os:translateToOWLIndividual refName="translationToWeatherForecast">
<os:elementForOWLIndividual type ="xsd:string"> $DailyWeatherForecastEvent </os:elementForOWLIndividual>
<os:IDForOWLIndividual type="xsd:string"> $DailyWeatherForecastEvent/ID </os:IDForOWLIndividual>
<os:OWLClass"> http://www.es.si/domainOntologies/ElectricityConsumption.owl #WeatherForecast </os:OWLClass>
<os:OWLClass"> http://www.es.si/domainOntologies/ElectricityConsumption.owl #DailyWeatherConditions </os:OWLClass>
</os:translateToOWLIndividual> <os:translateToOWLDataProperty> <os:referenceToTranslationFromIndividual
name="translationToWeatherForecast"> <os:OWLDatatypeProperty> http://www.es.si/domainOntologies/
ElectricityConsumption.owl#hasTemp </os:OWLDatatypeProperty>
<os:datatypeValue type="xsd:int"> $DailyWeatherForecastEvent/temperature </os:datatypeValue>
</os:translateToOWLDataProperty>
<os:translateToOWLDataProperty> <os:referenceToTranslationFromIndividual name="translationToWeatherForecast">
<os:OWLDatatypeProperty> http://www.es.si/domainOntologies/ElectricityConsumption.owl#forDay </os:OWLDatatypeProperty>
<os:datatypeValue type="xsd:int"> $DailyWeatherForecastEvent/forDate </os:datatypeValue> </os:translateToOWLDataProperty>
```

Furthermore, the translation should define whether data should be translated into the domain or event OWL KB. The main factor that influences this decision is whether the information that these data carry should be stored in the KB for a longer time and will be needed to support other activities, or if it is only required to detect CEs. In the former case, it should be translated into the domain KB; and in the latter case, it should be translated into the event KB. In the run-time, when an event is passed to the CES, the Complex Event Service Manager extracts event data, retrieves relevant event KB, and then the Translator maps its payload data into OWL. The mapping is based on translations that are defined for its event type. For definition of translations our approach adapts the JXML2OWL translation approach 14. Example translations for the sales report event type are given in Table 1.

Reasoner

Reasoner is the component that infers upon the OWL KB in order to derive new information. In our framework, the main purpose of this is to determine whether a CE has occurred or not. If all the CE definitions have been appropriately defined and if member event data is correctly translated into OWL, a new CE is detected when an individual is inferred to be a member of a CE class and has the value of its *trigger* property "true". This is an indicator for the event ontology handler that a CE should be announced by the CES. There are several available reasoners that can be used and support OWL, SWRL or both; examples are HermIT, Fact++, Pellet, and RacerPro.

Event ontology handler

Event ontology handler is responsible for actions upon event ontologies that are required in order to maintain it in a consistent state and up-to-date. It is also responsible for other activities, such as optimized reasoner invocations and merging existing OWL ontologies with translated OWL document. When an event is passed to the CES, the Event ontology handler activates those event ontologies in which the event that was caught is a member event of an event defined in the ontology. Every event ontology can be active or inactive. A trigger that defines when an event ontology should become active and processed has to be defined. Such triggers are represented by an occurrence of an event of a certain type. When such event occurs the corresponding event ontologies become active. An active OWL ontology is used as a part of the OWL KB. When it becomes active, individuals of every CE subclass are added to the KB. A CE type definition may address one or more member events of the same event type. For every occurrence of an event that the event ontology is subscribed to, an individual is added to the corresponding event OWL class. When inference by the Reasoner results in the trigger property value "true" for an individual that is a member of the ComplexEvent class, then the Complex event service manager triggers the complex event(s) and the Event ontology handler removes the complex event individual and member event individuals from the OWL KB. If there are no remaining individuals of the member event classes in the active OWL ontology, then the Event ontology handler returns the event ontology into the inactive state. Thus the event ontology handler handles OWL instances and does not act upon OWL's conceptual level. It maintains the basic CE instances and the *trigger* property values based on the actions it takes in response to event instances with the *trigger* property values *true*.

Conclusions and further work

In this paper we have presented a novel approach to defining and detecting complex events, which provides highly expressive complex event definitions using OWL ontologies extended with the SWRL. We have developed an EDA framework of the Complex Event Service that extends the service-oriented system architecture. This is especially important in environments with frequently changing business requirements. Our ontology-based framework for complex events comprises the design and run-time aspects of complex events. Complex event types are defined in design time with the ontology representation language OWL. At run-time, our CES reacts on incoming member events, infers and triggers complex events. The case study from the domain of electrical distribution has shown several important advantages of our framework. Firstly, it supports definition of complex event types with semantic expressivity requirements of various complexities. Secondly, the framework supports hierarchical definitions of complex event types. A complex event type can have several specializations, for example each with specific additional conditions. This allows for greater flexibility and allows for working with different levels of detail. Based on the information that is available, more or less specific event can be triggered. Furthermore, our framework integrates event and domain information, thus enabling to perform more complete and accurate inferences about complex event occurrences. In our case study, before our framework was applied, most of the member events had to be monitored manually to determine if a complex event has occurred. The reason for this was that most of the required complex events were not supported by other available systems. Our framework contributed to this situation and provided automatic support for complex events. The human resources that previously supported complex events could be reassigned. Ontologies have served as a very appropriate mechanism to support complex event types where semantics plays an important role and where domain information has to be taken into consideration. If numerical calculations are required to determine if a complex event has occurred, our framework can be used as a complement to other existing EDA and CEP approaches. Our framework is defined in a way, that supports such integration on the EDA level through event enabled ESB.

In our further work, we intend to extend the framework into a methodology that integrates the complex event development support with the business process modeling and execution.

References

1. **Adaikkalavan R., Chakravarthy S.** Event Specification and Processing for Advanced Applications: Generalization and Formalization // Proceedings of the International Conference on Database and Expert Systems Applications. – Regensburg, Germany, 2007. – P. 369–379.
2. **Briassouli A., Dasiopoulou S., Kompatsiaris I.** Ontology–Based Trajectory Analysis for Semantic Event Detection // Proceedings of the International Conference on Semantic Computing. – Irvine, California, USA, 2007. – P. 735–742.

3. **Bugaite D., Vasilecas O.** Events Propagation from the Business System Level into the Information System Level. – Barry C. et al. (eds.), Information Systems Development. – Springer US, 2009. – 252 p.
4. **Chandy K. Schulte, W.** Event Processing: Designing IT Systems for Agile Companies. – McGraw Hill, 2009. – 256 p.
5. **Cheng S.–y., Jih W–r., Hsu J. Y.–j.** Context–aware Policy Matching in Event–Driven Architecture // Proceeding of the AAAI 2005 Workshop: Contexts and Ontologies: Theory, Practice and Applications. – Pittsburgh, Pennsylvania, USA, 2005. – P. 140–141.
6. **Francois A. R. J., Nevatia R., Hobbs J., Bolles R. C., Smith J. R.** VERL: an ontology framework for representing and annotating video events // IEEE Multimedia. – IEEE Computer Society Press, 2005. – No. 12(4). – P. 76–86.
7. **Gruber T. R.** Toward principles for the design of ontologies used for knowledge sharing // International Journal of Human–Computer Studies. – Academic Press, Inc., 1995. – No. 43(4–5). – P. 907–928.
8. **Juric. M, Loganathan R., Poornachandra S., Jennings F.** SOA approach to integration. – Packt Publishing, 2008. – 384 p.
9. **Kuusik A., Reilent E., Lõõbas I., Luberg A.** Data Acquisition Software Architecture for Patient Monitoring Devices // Electronics and Electrical Engineering. – Kaunas: Technologija, 2010. – No. 9(105). – P. 97–100.
10. **Luckham D. C., Schulte R.** Event Processing Glossary – Version 1.1. – 2008. Online: http://complexevents.com.
11. **Luchham D. C.** The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. – Addison–Wesley Longman Publishing Co., 2001. – 400 p.
12. **Moser T., Roth H., Rozsnyai S., Mordinyi R., Biffl S.** Semantic Event Correlation Using Ontologies // Proceedings of the 8th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE'2009). – Algarve, Portugal, 2009. – LNCS 5871, Springer–Verlag. – P. 1087 – 1094.
13. **Paschke A.** A Semantic Design Pattern Language for Complex Event Processing // Proceedings of Intelligent Event Processing – AAAI Spring Symposium 2009. – Stanford, USA, 2009. – P. 54 – 60.
14. **Rodrigues T., Rosa P., Cardoso J.** Moving from syntactic to semantic organizations using JXML2OWL // Computers in Industry. – Elsevier Science Publishers, 2009. – No. 59(8). – P. 808–819.
15. **Rupnik R.** Decision support system to support the solving of classification problems in telecommunications // Informacije MIDEM. – MIDEM, Society for Microelectronics, Electronic Components and Materials, 2009. – No. 39(3). – P. 168–177.
16. **Rupnik R., Kukar M., Krisper M.** Integrating data mining and decision support through data mining based decision support system // The Journal of Computer Information Systems. – IACIS, 2007. – No. 47(3). – P. 89–104.
17. **Sasa A., Juric M., Krisper M.** Service–Oriented Framework for Human Task Support and Automation // IEEE Transactions on Industrial Informatics. – IEEE, 2008. – No. 4(4). – P.292–302.
18. **Sen S., Ma J.** Contextualised Event–driven Prediction with Ontology–based Similarity // Proceedings of the 2009 AAAI Spring Symposium, Intelligent Event Processing. – The AAAI Press, Stanford University, CA, USA, 2009. – P. 73–79.
19. **Taylor H., Martinez F., Yochem A., Phillips L.** Event–Driven Architecture: How SOA Enables the Real–Time Enterprise. – Addison–Wesley, 2009. – 272 p.
20. **Teymourian K., Paschke A. Teymourian K., Paschke A.** Semantic Rule–Based Complex Event Processing. Rule Interchange and Applications // Lecture Notes in Computer Science. – SpringerLink, 2009. – Vol. 5858/2009. – P. 82–92.
21. **Turcu C., Prodan R., Cerlincă M., Turcu C., Cerlincă T.** RFID@B2B a Powerful Enabler of Business Transformation // Electronics and Electrical Engineering. – Kaunas: Technologija, 2009. – No. 5(93). – P. 59–64.
22. **Vukmirović S., Erdeljan A., Lendak I., Čapko D.** Extension of the Common Information Model with Virtual Meter // Electronics and Electrical Engineering. – Kaunas: Technologija, 2011. – No. 1(107). – P. 59–64.

Received 2011 04 17

A. Sasa, O. Vasilecas. Ontology-Based Support for Complex Events // Electronics and Electrical Engineering. – Kaunas: Technologija, 2011. – No. 7(113). – P. 83–88.

With increasing demands for agility and reactivity, scientific circles and leading ICT companies have recognized the need for EDA (event-driven architecture) and complex event processing in factory automation, signal processing, data acquisition, manufacturing, and other applications of industrial electronics. We propose an EDA framework for defining and processing complex events based on ontologies. The main novelty of our approach is introduction of semantic aspects into EDA. Our framework enables highly expressive semantic definitions of complex events and a higher level of semantic integration of real-time events with the domain and event ontologies. The main component of the framework is the complex event service that acts as an event sink for member events and as an event source for inferred complex events. Our approach is complementary to existing approaches and can enhance their support for complex events that require higher expressivity and semantic descriptions. Ill. 4, bibl. 22, tabl. 1 (in English; abstracts in English and Lithuanian).

A. Šaša, O. Vasilecas. Ontologija grindžiamas sudėtinių įvykių palaikymas // Elektronika ir elektrotechnika. – Kaunas: Technologija, 2011. – Nr. 7(113). – P. 83–88.

Kadangi lanksčiai adaptyvių ir reaktyvių sistemų savybių paklausa nuolat didėja, mokslininkų grupės ir svarbiausios IKT kompanijos pripažino įvykiais valdomos architektūros (EDA) ir sudėtinių įvykių apdorojimo poreikį įmonių automatizavimo, signalų apdorojimo, duomenų surinkimo ir kaupimo, gamybos, ir kitose pramoninės elektronikos taikymo srityse. Straipsnyje pasiūlytas ontologija grindžiamas EDA sudėtinių įvykių apibrėžimo ir apdorojimo karkasas. Pasiūlytojo karkaso naujumas pirmiausia yra susijęs su semantinių aspektų įdiegimu EDA. Karkasas įgalina apibrėžti aukštesnio ekspresyvumo lygmens realaus laiko sudėtinių įvykių apibrėžimą ir aukštesnį realaus laiko semantinės integracijos lygį naudojant dalykinės srities ir įvykių ontologiją. Pagrindinis karkaso komponentas yra sudėtinių įvykių paslauga, kuri veikia kaip sudėtinio įvykio atskirų narių išnykimo vieta ir kaip išvestinių sudėtinių įvykių šaltinis. Straipsnyje pasiūlytas būdas papildo esamus sudėtinių įvykių apdorojimo būdus ir gali sustiprinti bei pagerinti jų teikiamą sudėtinių įvykių palaikymą, tais atvejais, kai reikalingas aukštesnio lygio ekspresyvumas ir semantinis apibūdinimas. Il. 4, bibl. 22, lent. 1 (anglų kalba; santraukos anglų ir lietuvių k.).