MEdit4CEP-CPN: An approach for Complex Event Processing modeling by Prioritized Colored Petri Nets

Juan Boubeta-Puig a,*, Gregorio Díaz b, Hermenegilda Macià b, Valentín Valero b, Guadalupe Ortiz d

a Department of Computer Science and Engineering, University of Cádiz, Avda. de la Universidad de Cádiz 10, Puerto Real, Cádiz, 11519, Spain
b School of Computer Science, University of Castilla-La Mancha, Campus Universitario s/n, Albacete 02071, Spain

d Additional author.

ARTICLE INFO

Article history:
Received 3 January 2017
Revised 6 November 2017
Accepted 21 November 2017
Available online xxx

Keywords:
Formal modeling
Petri Nets
Event-Based System
Complex Event Processing
Event Processing Language
Model-driven engineering
Business process management

ABSTRACT

Complex Event Processing (CEP) is an event-based technology that allows us to process and correlate large data streams in order to promptly detect meaningful events or situations and respond to them appropriately. CEP implementations rely on the so-called Event Processing Languages (EPLs), which are used to implement the specific event types and event patterns to be detected for a particular application domain. To spare domain experts this implementation, the MEdit4CEP approach provides them with a graphical modeling editor for CEP domain, event pattern and action definition. From these graphical models, the editor automatically generates a corresponding Esper EPL code. Nevertheless, the generated code is syntactically but not semantically validated. To address this problem, MEdit4CEP is extended in this paper by Prioritized Colored Petri Net (PCPN) formalism, resulting in the MEdit4CEP-CPN approach. This approach provides both a novel PCPN domain-specific modeling language and a graphical editor. By using model transformations, event pattern models can be automatically transformed into PCPN models, and then into the corresponding PCPN code executable by CPN Tools. In addition, by using PCPNs we can compare the expected output with the actual output and can even conduct a quantitative analysis of the scenarios of interest. To illustrate our approach, we have conducted an air quality level detection case study and we show how this novel approach facilitates the modeling, simulation, analysis and semantic validation of complex event-based systems.

© 2017 The Authors. Published by Elsevier Ltd.
This is an open access article under the CC BY-NC-ND license. (http://creativecommons.org/licenses/by-nc-nd/4.0/)

1. Introduction

Complex Event Processing (CEP) [1] provides users with facilities for analyzing and correlating large volumes of data in the form of events enabling them to detect relevant or critical situations for a particular domain in real time. To meet this objective, the conditions describing the situations of interest must be specified as event patterns. Patterns are implemented by using the languages provided by CEP engines, the so-called Event Processing Languages (EPLs), and once the patterns are defined they can be deployed in the CEP engine in question.

One of the main advantages of CEP is the diversity of scenarios and domains which can benefit from its use. However, there is a common handicap: event patterns should be defined by domain experts, but most domain experts are not proficient in programming languages and programmers are not skilled in programming the patterns for unknown domains. To solve this problem, we successfully proposed MEdit4CEP [2], a model-driven solution for real-time decision making in Event-Driven Service-Oriented Architecture (SOA 2.0) [3]. This solution consists of a graphical modeling editor for CEP domain definition and a graphical modeling editor for event pattern and action definition. Code is then automatically generated from these graphical models. In addition to automatic code generation, MEdit4CEP offers two key features: on the one hand, domain experts can graphically model domains and patterns; on the other hand, the event pattern editor can automatically be reconfigured for different CEP domains, which will have been previously modeled by domain experts.

Therefore, domain experts have the support of a graphical tool to define event types and event patterns, avoiding the need to learn any programming language. The EPL pattern code is then au-

* Corresponding author.
E-mail addresses: juan.boubeta@uca.es (J. Boubeta-Puig), gregorio.diaz@uclm.es (G. Diaz), hermenegilda.maci@uclm.es (H. Macià), valentin.valero@uclm.es (V. Valero), guadalupe.ortiz@uca.es (G. Ortiz).

https://doi.org/10.1016/j.is.2017.11.005
0306-4379/© 2017 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license. (http://creativecommons.org/licenses/by-nc-nd/4.0/)

Please cite this article as: J. Boubeta-Puig et al., MEdit4CEP-CPN: An approach for Complex Event Processing modeling by Prioritized Colored Petri Nets, Information Systems (2017), https://doi.org/10.1016/j.is.2017.11.005
tomatically generated from the graphical models and deployed in the subsequent CEP engine. MEdit4CEP provides a syntactic validation of the patterns designed, so we can ensure that the generated code is syntactically correct. However, MEdit4CEP does not provide semantic validation.

In order to provide a semantic validation, we worked on the transformation of CEP operators into Prioritized Colored Petri Nets (PCPN) [4], generating the transformations from a number of Esper EPL operators [5] to PCPN, which were validated through a case study.

In this paper, we go one step further by proposing MEdit4CEP-CPN, an extension of MEdit4CEP by means of the PCPN formalism, which provides a Domain-Specific Modeling Language (DSL) and a graphical editor for automatically transforming event pattern models into PCPN graphical models. These models are then validated and transformed into code executable by Petri Nets software.

Our DSL is composed of a PCPN metamodel and validation rules, which can be used to edit and syntactically validate PCPN tool-independent models. The DSL also provides the transformation rules required to automatically transform a model conforming to Model4CEP [6] (our event pattern metamodel for defining patterns in a user-friendly way) into a model conforming to the proposed PCPN metamodel. This transformation to PCPN is based on the mapping defined in the previous work [4], with improvements in order to cover the concatenation of patterns.

In addition, the DSL provides a set of model-to-text transformation rules so that the models conforming to the PCPN metamodel are now automatically transformed into input code for CPN Tools. We can thus utilize all the CPN Tools functions to analyze the PCPN obtained. Specifically, we can obtain results from simulations in order to check the expected pattern behavior, but we can also perform a quantitative analysis of the PCPN so as to generate predictions in hypothetic scenarios.

Thus, the mapping of EPL to PCPN, which is a formalism often used in Business Process Management (BPM) analysis [7,8], provides us with a unified framework for the analysis of event-driven business processes by integrating Event-Based Systems (EBSs), CEP and process control-flow operations in an all-in-one solution. On the one hand, the integration of a CEP engine with an EBS, characterized by connecting event producers and event consumers through a message-oriented publication/subscription middleware, enables us to correlate and aggregate events in order to discover and respond to event patterns. On the other hand, combining existing techniques for CEP and BPM allows us to both monitor the execution of business processes and analyze finished process instances [9]. In this way, the EBSs are suitable for managing events created by processes.

The contributions of this paper are, therefore: (1) a DSL for defining PCPN models and validating and transforming them into code executable by software supporting Petri Nets; (2) a graphical editor for this DSL, which allows the user to graphically model the PCPNs and to proceed with all the transformation and validation steps through the editor; and (3) the integration of this editor with MEdit4CEP. As a result, end users can now execute the following steps using MEdit4CEP-CPN: (1) they can graphically model CEP patterns for the domain in question; (2) they can validate the pattern syntax; (3) they can then automatically transform the graphical patterns into a PCPN model, which can also be edited by using our graphical editor; (4) they can validate the PCPN model syntax; (5) subsequently, they can automatically generate code for Petri Nets software; (6) they can validate the pattern semantics and address quantitative analysis; and (7) they can finally automatically generate the EPL code and deploy it in the final system architecture. Until now, only steps (1), (2) and (7) were possible; with the contributions of this paper we achieve steps (3)–(6), which have been properly integrated with the existing functionalities.

Our approach has been validated through a case study for air pollutant monitoring. This case study exemplifies how the event patterns for detecting air quality changes are (1) graphically modeled and syntactically validated, (2) automatically converted into PCPN and (3) semantically validated and quantitatively analyzed.

The rest of the paper is organized as follows. Section 2 explains the required background on CEP, MEdit4CEP and PCPN to facilitate the understanding of this paper. Section 3 presents the model-driven approach in a nutshell, from the graphical definition of patterns to the generation and analysis of the corresponding PCPNs by using CPN Tools, and the generation of their EPL implementation code. The PCPN DSL and graphical editor are then presented in Section 4. Section 5 describes the PCPN modeling for the CEP operators required in the Air Quality Level (AQL) case study, which is then presented and analyzed in Section 6. Relevant related works are detailed in Section 7 and, finally, Section 8 draws some conclusions and future research lines.

2. Background

This section first provides a background explanation on CEP technology. Then, MEdit4CEP is described. Finally, the PCPN formalism we use for modeling CEP-based systems is explained.

2.1. Complex Event Processing

CEP [10] is a technology which emerged recently with the aim of detecting relevant or critical situations (complex events) in real time. It is used to analyze and correlate huge amounts of data in the form of events. A situation is an event occurrence or an event sequence that requires an immediate reaction [10]. Within this scope, events can be classified into two main categories: simple events are those which are indivisible and happen at one point in time, while complex events provide additional semantic significance, which summarizes a set of other previous events [11]. In fact, events can be derived from other events by matching the so-called event patterns, that is, templates which describe the conditions required to deal with a situation. These patterns are implemented by using EPLs, which can be classified into the following language styles [10]: stream-oriented, rule-oriented and imperative. Further information about existing EPLs can be found in the survey by Cugola and Margara [12].

In this context, a software capable of matching these patterns over continuous and heterogeneous event streams is required. This software is a CEP engine, which is also in charge of raising real-time alerts when a pattern is met.

Remarkably, we have chosen to use Esper engine, and therefore Esper EPL, for a number of reasons. First, Esper is a highly efficient open-source CEP engine: it can process over 500,000 events/s [13]. Second, Esper EPL is a rich high-level processing language which is more complete than other EPL languages. Furthermore it supplies a wider range of temporal and pattern operators for the definition of situations of interest. For the sake of brevity, we refer to Esper EPL simply as EPL throughout the rest of the paper. Moreover, Esper provides the Esper EPL online tool [14] for simulating events as well as implementing and detecting event patterns.

CEP is performed in 3 stages, as depicted in Fig. 1:

1. Event capture –this consists of receiving events which will later be analyzed using CEP technology.
2. Analysis –this stage involves processing and correlating the information in the form of events according to the previously defined patterns in order to detect critical or relevant situations in real time.
3. Response –this refers to notifying the system, software or device in question when detecting a particular situation of interest.

Please cite this article as: J. Boubeta-Puig et al., MEdit4CEP-CPN: An approach for Complex Event Processing modeling by Prioritized Colored Petri Nets, Information Systems (2017), https://doi.org/10.1016/j.is.2017.11.005
دریافت فوری متن کامل مقاله

امکان دانلود نسخه تمام متن مقالات انگلیسی
امکان دانلود نسخه ترجمه شده مقالات
پذیرش سفارش ترجمه تخصصی
امکان جستجو در آرشیو جامعی از صدها موضوع و هزاران مقاله
امکان دانلود رایگان ۲ صفحه اول هر مقاله
امکان پرداخت اینترنتی با کلیه کارت های عضو شتاب
دانلود فوری مقاله پس از پرداخت آنلاین
پشتیبانی کامل خرید با بهره مندی از سیستم هوشمند رهگیری سفارشات