Expert Systems with Applications 41 (2014) 445–456

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

A model-driven approach for facilitating user-friendly design of complex event patterns

Juan Boubeta-Puig *, Guadalupe Ortiz, Inmaculada Medina-Bulo

Department of Computer Science and Engineering, University of Cádiz, C/ Chile 1, 11002 Cádiz, Spain

ARTICLE INFO

Keywords:
Complex event processing
Model-driven development
Event processing language
Fast data

ABSTRACT

Complex Event Processing (CEP) is an emerging technology which allows us to efficiently process and correlate huge amounts of data in order to discover relevant or critical situations of interest (complex events) for a specific domain. This technology requires domain experts to define complex event patterns, where the conditions to be detected are specified by means of event processing languages. However, these experts face the handicap of defining such patterns with editors which are not user-friendly enough. To solve this problem, a model-driven approach for facilitating user-friendly design of complex event patterns is proposed and developed in this paper. Besides, the proposal has been applied to different domains and several event processing languages have been compared. As a result, we can affirm that the presented approach is independent both of the domain where CEP technology has to be applied to and of the concrete event processing language required for defining event patterns.

1. Introduction

In recent years, volumes of data produced by a variety of heterogeneous sources have increased around the world (Tsuchiya, Sakamoto, Tsuchimoto, & Lee, 2012). As a result, both *Information Technology* (IT) and business users need to efficiently collect and process this huge amount of data in real time to discover relevant situations which will allow driving successful business decisions or actions (Hansen, 2013).

In this regard, *big data* is an approach which helps to process this huge amount of data. It is characterized in terms of the three *V's*: *Volume*, *Velocity* and *Variety* (Russom, 2011). Volume refers to the amount of data that can be managed and stored every day. Velocity is the big data dimension which deals with measuring how fast data can be collected and analyzed. Variety means the different existent data types: audio, video, text, etc. However, big data normally focus on data previously collected and stored in databases. For that reason, it is not the best solution to process data from different sources in real time. To solve it, big data can be complemented with *fast data* (Hansen, 2013), an approach which allows to continuously analyze data and which can be characterized by a new dimension known as *Value*. This dimension aims to determine why such data is important for business.

In order to detect relevant or critical situations in business, fast data may be integrated with *Complex Event Processing* (CEP) (Luckham, 2002), a technology that allows detecting meaningful events in real time and inferring valuable knowledge for end users. For that purpose, the conditions describing the situations to be detected must be specified by using special templates known as *event patterns*. These patterns will be added into an event processing engine, the software responsible for analyzing and correlating the events received from different sources, as well as for raising alerts to users or systems interested in *complex events* (situations) generated by the detected event patterns.

These event patterns are defined using specific languages — developed for this purpose— known as *Event Processing Languages* (EPLs). Nevertheless, a wide experience on EPLs is required for defining such patterns. Thereby, one of the main drawbacks of using CEP by non-technical users, who are the ones having the domain-specific knowledge on the pattern to be detected, is the big learning curve necessary for becoming an expert in these languages. Some software solutions, such as Esper's editor (EsperTech Inc., 2013), Oracle CEP Visualizer (Oracle, 2013), StreamBase Studio (StreamBase, 2013) and SAP Sybase ESP Studio (Sybase, 2013), provide graphical tools to address this problem. Despite this fact, these tools are not user-friendly enough since non-experts on CEP have to write some EPL code by hand.

As a solution, in this paper, we propose a model-driven approach so that domain experts (but non experts on CEP) can concentrate in the graphical definition of event patterns in a user-friendly way without the need of hand-writing any code. Afterwards, the required code will be automatically generated. In concrete, our approach has four major contributions. Firstly, a metamodel is proposed to define event patterns as models which

* Corresponding author. Tel.: +34 956 01 56 92.
E-mail addresses: juan.boubeta@uca.es (J. Boubeta-Puig), guadalupe.ortiz@uca.es (G. Ortiz), inmaculada.medina@uca.es (I. Medina-Bulo).

0957-4174/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2013.07.070

do not depend on the specific EPL required by the final engine used for complex event processing. Secondly, a domain-independent editor is implemented from this metamodel to facilitate user-friendly design of event patterns. Thirdly, a model validation process checks the correctness of these event patterns represented as models. Fourthly, a model transformation process automatically transforms such models into any particular EPL —Esper EPL code in this work. Furthermore, a case study in the field of health care is described and implemented for illustrating our proposal, which is also evaluated and discussed.

The rest of this paper is organized as follows. Section 2 includes background on *Model-Driven Software Development* (MDSD), CEP and EPL. Section 3 describes our model-driven approach in a nutshell and, afterwards, this approach is detailed in the following sections. In concrete, Section 4 explains the EPL metamodel for defining event patterns, Section 5 describes the implemented editor, Section 6 specifies the metamodel constraints and Section 7 details the process for transforming event pattern models into EPL code and the latter integration into a CEP engine. Then, Section 8 describes the application of our approach in a health-care case study. Subsequently, our approach is evaluated and discussed in Section 9. Some related works are described in Section 10. Finally, the conclusion and future work are highlighted in Section 11.

2. Background

In this section, the relevant subject matters for the scope of this paper, MDSD, CEP and EPL, are introduced.

2.1. Model-driven software development

MDSD is an important paradigm in software development which aims to find domain-specific abstractions and make them accessible by means of formal modeling (Stahl, Voelter, & Czarnecki, 2006). These abstract representations of aspects of a system, known as *models*, are used as primary artifacts in the development process (Hussmann, Meixner, & Zuehlke, 2011). The key features of this paradigm is that makes use of models of different levels of abstraction and provides *model transformations* in order to automatically transform a model into another as well as a model into implementation code.

Each model is an instance of a *metamodel*. In this scope a metamodel describes the structure of models in an abstract way. Particularly, a metamodel is defined using a metamodel language joined to a set of rules which specify the constraints so that the metamodel is well-formed. The most well-known metamodel language is Ecore and the *de facto* standard for capturing such constraints is *Object Constraint Language* (OCL).

This way, MDSD facilitates the automation of software production, increasing the productivity, quality and maintainability of software systems (Stahl et al., 2006). Even more, domain experts (non-technical users) can also understand such models, so that they can play an active role in software development.

2.2. Complex event processing

CEP is a cutting-edge technology which provides powerful techniques for processing and correlating events in order to detect relevant or critical business situations (complex events) in real time.

An event can be defined as anything that happens or could happen (Luckham, 2012). Mainly, events can be classified into three categories: a *simple event* is indivisible and happens at a point in time, a *complex event* contains more semantic meaning which summarizes a set of other events, and a *derived event* is generated when applying a process to one or more other events (Event Processing Technical and Society, 2011). Events can be derived from other events by applying or matching *event patterns*, templates where the conditions describing the situations to be detected are specified. A CEP engine is the software used to match these patterns over continuous and heterogenous event streams (timely ordered sequence of events of multiple types), and to raise alerts about the complex events created when detecting such event patterns.

According to Vincent (2010), *CEP systems*, as well as other decision-support systems such as *expert systems* take expert event-driven decisions, where expert knowledge is encoded from the available subject matter experts. In addition, these systems use "rules" (or event patterns) to determine whether stated goals (conditions) are fulfilled.

CEP can be applied to different areas. According to Luckham (2012), some of the major areas for sales of CEP are: fraud detection and security (Edge & Falcone Sampaio, 2012), transportation and traffic management (Dunkel, Fernández, Ortiz, & Ossowski, 2011), health care (Yuan & Lu, 2009; Yao, Chu, & Li, 2011), energy and manufacturing (Vikhorev, Greenough, & Brown, 2013), location-based services (Uhm, Lee, Hwang, Kim, & Park, 2011), financial systems and operations (Edge & Falcone Sampaio, 2012), and operational intelligence in business (Chaudhuri, Dayal, & Narasayya, 2011). Among other additional areas, CEP can also be applied to home automation (Romero et al., 2011) and RFID signals (Yao et al., 2011).

To sum up, CEP allows detecting meaningful events and inferring valuable knowledge for end users in different domains. The main advantage of using this technology to process complex events is that the latter can be identified and reported in real time, reducing the latency in decision making, unlike the methods used in traditional software for event analysis.

2.3. Event processing language

As previously mentioned, in order to detect *situations of interests* on specific areas it is necessary the definition of so-called *event patterns*. These event patterns are defined using specific languages developed for this purpose known as EPLs. According to Etzion and Niblett (2010), these languages can be classified by the following language styles: stream-oriented, rule-oriented and imperative.

Stream-oriented EPLs are SQL-like languages but including new concepts, such as timing and temporal relationships. The learning curve is not high because their syntax is very close to SQL, worldwide known. Some of these EPLs are: Esper EPL (EsperTech Inc., 2013), CQL (Oracle, 2013), StreamSQL (StreamBase, 2013) and CCL (Sybase, 2013). In this work, we decided to transform graphical event patterns into Esper EPL since this language provides more operators than the others and its open-source engine is very efficient: it can process over 500,000 events/s (EsperTech Inc., 2013).

Rule-oriented EPLs implement event queries where condition expressions are evaluated over a set of facts. Some of CEP solutions that provide rule-oriented EPLs are: IBM Operational Decision Management (IBM, 2013), Drools Fusion (JBoss, 2013) and ETALIS (ETALIS, 2013).

Imperative EPLs define rules in an imperative way where operators define transformations over their inputs. Progress Apama (Progress Software, 2013) is an event processing platform which provides this EPL style.

Further information about other existing EPLs and CEP systems can be found in the survey by Cugola and Margara (2012).