ER Designer Toolkit: A Graphical Event Definition Authoring Tool

Pythagoras Karampiperis Giannis Mouchakis George Paliouras Vangelis Karkaletsis

Software and Knowledge Engineering Laboratory, Institute of Informatics and Telecommunications, National Center for Scientific Research "Demokritos"
Agia Paraskevi Attikis, P.O.Box 60228, 15310 Athens, Greece

pythk@iit.demokritos.gr gmouchakis@gmail.com paliourg@iit.demokritos.gr vangelis@iit.demokritos.gr

ABSTRACT
Currently there exist several tools for Complex Event Recognition, varying from design platforms for business process modeling (BPM) to advanced Complex Event Processing (CEP) engines. Several efforts have been reported in literature aiming to support domain experts in the process of defining event recognition (ER) rules. However, few of them offer graphical design environments for the definition of such rules, limiting the broad adoption of ER systems. In this paper, we present a graphical Event Definition Authoring Tool, referred to as the Event Recognition Designer Toolkit (ERDT) with which, a domain expert can easily design event recognition rules on temporal data and produce standalone Event Recognizers.

Categories and Subject Descriptors
H.4 [**Information Systems Applications**]: Miscellaneous; I.5 [**Pattern Recognition**]: Miscellaneous

General Terms
Design, Documentation.

Keywords
Authoring Tools, Complex Event Processing, Event Recognition.

1. INTRODUCTION
Today's organisations are able to collect data in various structured and unstructured digital formats, but they do not have the capability to fully utilise these data to support and improve their resource management process. It is evident that the analysis and interpretation of the collected data needs to be automated and transformed into operational knowledge.

PETRA'11, May 25 - 27, 2011, Crete, Greece.

Events are particularly important pieces of knowledge, as they represent the temporal nature of the processes taking place in an organisation. Therefore, the recognition of events is of outmost importance in resource management [7].

Currently there exist several tools for Complex Event Recognition, varying from design platforms for business process modeling (BPM) to advanced Complex Event Processing (CEP)

engines. Several efforts have been reported in literature aiming to support domain experts in the process of defining event recognition (ER) rules. However, few of them offer graphical design environments for the definition of such rules.

In this paper, we present a graphical Event Definition Authoring Tool with which, a domain expert can easily design event recognition rules on temporal data and produce standalone Event Recognizers.

The paper is structured as follows: First, we discuss the authoring tools available in literature, which offer graphical design environments. Then, we discuss the design requirements towards a graphical authoring tool that supports domain experts in the process of defining event recognition (ER) rules, and present the architecture of our proposed tool. Finally, we discuss the technologies used in the development of our tool and demonstrate how the proposed tool can be used in practice.

2. ER AUTHORING TOOLS: A SHORT SURVEY
In the literature, few systems have been proposed offering graphical design environments for the definition of ER rules. Below, we review these environments:

The *Aleri Streaming Platform* [1] is a system that offers a simple graphical language to define event processing rules, by combining a set of predefined operators. To increase the system's expressiveness, custom operators can be defined using a scripting language called Splash, which includes the capability of defining variables to store past information items, so that they can be referenced for further processing. Pattern detection operators are provided as well, based on sequences. Pattern matching can take place in the middle of a complex computation, and sequences may use various attributes for ordering, other than timestamps. As a consequence, the semantics of output ordering does not necessarily reflect timing relationships between input items.

The platform is designed to scale by exploiting multiple cores on a single machine or multiple machines in a clustered environment. However, no information is provided on the protocols used to distribute operators. Interestingly, the Aleri Streaming Platform is designed to easily work together with other business instruments: probably the most significant example is the Aleri Live OLAP system, which extends traditional OLAP solutions [2] to provide near real-time updates of information. Additionally, Aleri provides adapters to enable different data formats to be translated into flows of items compatible with the Aleri Streaming Platform, together with an API to write custom programs that may interact with the platform using either standing or one-time queries.

StreamBase [8] is a software platform that includes a data stream processing system, a set of adapters to gather information from heterogeneous sources, and a developer tool based on Eclipse. It uses a declarative, SQL-like language for rule specification, called StreamSQL [9]. Besides traditional SQL operators, StreamSQL offers customizable time-based and count-based windows. Additionally, it includes a simple pattern-based language that captures conjunctions, disjunctions, negations, and sequences of items. Operators defined in StreamSQL can be combined using a graphical plan-based rule specification language, called EventFlow. User-defined functions, written in Java or C++, can be added as custom aggregates.

Oracle CEP [5] is an event-driven architecture suite embedding BEA's WebLogic Event Server. It provides real-time information flow processing, and uses CQL as its rule definition language. Similar to StreamBase, it adds a set of relation-to-relation operators designed to provide pattern detection, including conjunctions, disjunctions, and sequences. An interesting aspect of this pattern language is the possibility for users to program the selection and consumption policies of rules.

Like in StreamBase, a visual, plan-based language is also available inside a development environment based on Eclipse. This tool enables users to connect simple rules into a complex execution plan. Oracle CEP is integrated with existing Oracle solutions, which includes technology for distributed processing in clustered environment, as well as tools for analysis of historical data.

Tibco Business Events [10] is another widespread complex event processing system. It is mainly designed to support enterprise processes and to integrate existing Tibco products for business process management. To do so, Tibco Business Events exploits the pattern-based language of Rapide, which enables the specification of complex patterns to detect occurrences of events and the definition of actions to automatically react after detection. Interestingly, the architecture of Tibco Business Events is capable of decentralized processing, by defining a network of event processing agents: each agent is responsible for processing and filtering events coming from its own local scope.

Other widely adopted commercial systems exist, for which, unfortunately, documentation or evaluation copies were not available. We mention here some of them.

IBMWebSphere Business Events: IBM acquired AptSoft CEP system during 2008 and renamed it to WebSphere Business Events [3]. Today, it is fully integrated inside the WebSphere platform, which can be deployed on a clustered environment for faster processing. IBM WebSphere Business Events provides a graphical front-end, which helps users writing rules in a pattern-based language. Such a language allows detection of logical, causal, and temporal relationships between events, using an approach similar to the one described for Tibco Business Events.

Progress Apama Event Processing Platform: The Progress Apama Event Processing Platform [6] has been recognized as a market leader for its solutions and for its strong market presence [4]. It offers a development tool for rule definition, testing and deployment, and a high performance engine for detection.

3. DESIGN REQUIREMENTS FOR ER AUTHORING TOOLS

From the review of the ER Authoring Tools, we can observe that all of them use either a query-based or a rule-based ER language approach. Moreover each of them supports a single ER language.

Table 1. Features of the examined Authoring Tools

ER Authoring Tool	Query-based	Rule-based	User Operators	Multiple Lang.	Open Source	Cross-platform
Aleri	✓	-	-	-	-	
Apama	-	✓	N/A	-	-	-
IBM WSBE	-	✓	N/A	-	-	✓
OracleCEP	✓	-	✓	-	-	✓
Streambase	✓	-	✓	-	-	✓
Tibco	-	✓	-	-	-	✓

Furthermore, not all the tools support the definition of user-defined operators. As a result the flexibility in the design is limited. Although most systems are cross-platform, none of them is open source. Thus, their wide adoption is restricted by the business strategies of their vendors. These observations are summarized in Table 1.

As a result, our tool should meet the following design considerations:

- Provide a graphical user interface, that is simple and user friendly. This will help a domain expert to design event rules without having to be familiar with the details of the ER language in use.
- Provide the ability to generate ER rules using both query and rule-based ER languages. This can make the tool interoperable in many different ER platforms.
- Support user-defined operators, in order to increase the tool's flexibility.
- Be cross platform and open source, so that it can be shared with the community, maximizing its impact and possible extension.

4. AUTHORING TOOL ARCHITECTURE

By taking the above mentioned design considerations into account, we defined the architecture of the ER Authoring Tool.

The architecture of the tool has been specified so as to be adaptable to specific requirements resulting from the Event Recognition language in use, as well as, to be extendable to support new Event Recognition languages and user-defined Event Recognition building blocks (operators). Moreover, the tool uses cross-platform technologies, which maximize the potential of its use.

The architecture consists of the following main components, as depicted in Figure 1:

– *The Design Engine*. This is where the rules are created in the form of a directed acyclic graph of operators. The engine is the result of the combination of a Graphical Domain Model with a Validation Model. The Graphical Domain Model provides the engine with all the information needed to design a model. It uses a pool of available operators (constructs) and the Palette Model which provides the different figures for the representation of the constructs and the type of connections between them. The Validation Model is a set of integrity constraints that the generated graph should meet, e.g. the graph should be acyclic, two constructs cannot have the same name property etc.

– *The ER Rule Generator* which processes the directed acyclic graph and creates rules in an ER language-independent form. It can also extend the pool of the available constructs by transforming a rule to a new construct which can be used in the Design Engine.

The ER Language Compiler. The compiler uses an extendable pool of ER Language Libraries and transforms the generated rules into Event Recognizers that use the preferred ER language.

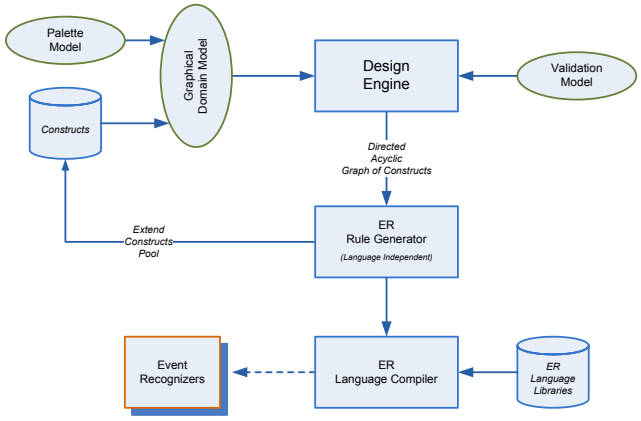

Figure 1. Authoring Tool Architecture

5. AUTHORING TOOL TECHNOLOGIES

The ER Authoring Tool was developed in Eclipse Helios 3.6.1 and is written in Java SE 6. We used the Eclipse Graphical Modeling Framework (GMF) which is a component of the Eclipse Modeling Project and uses the Eclipse Modeling Framework (EMF) and the Graphical Editing Framework (GEF). Below is a brief description of those technologies:

– *Graphical Modeling Framework* (GMF): Released as part of the Eclipse 3.2 Callisto in June 2006. It is a framework for building modeling-like graphical Eclipse-based editors such as UML editors, workflow editors, etc. The framework can be divided into two main components: the tooling and the runtime. The tooling consists of editors to create/edit models describing the notational, semantic and tooling aspects of a graphical editor, as well as a generator to produce the implementation of graphical editors. The generated plug-ins depend on the GMF Runtime component to produce a world class extensible graphical editor.

– *Graphical Editing Framework* (GEF): An open source framework which provides technology to create rich, consistent graphical editors and views for the Eclipse Workbench UI. It has been used to build a variety of applications, such as state diagrams, activity diagrams, class diagrams, GUI builders for AWT, Swing and SWT, and process flow editors. It bundles three components:

- Draw2d: A layout and rendering toolkit for displaying graphics on an SWT Canvas.
- GEF-MVC: An interactive model-view-controler (MVC) framework, which fosters the implementation of SWT-based tree and Draw2d-based graphical editors for the Eclipse Workbench UI.
- Zest: A visualization toolkit based on Draw2d, which enables implementation of graphical views for the Eclipse Workbench UI.

– *Eclipse Modeling Framework Project* (EMF): A modeling framework and code generation facility for building tools and other applications based on a structured data model. From a model specification described in XML, EMF provides tools and runtime support to produce a set of Java classes for the model, a set of adapter classes that enable viewing and command-based editing of the model, and a basic editor. Models can be specified using annotated Java, XML documents, or modeling tools like Rational Rose, which can then be imported into EMF. Most important of all, EMF provides the foundation for interoperability with other EMF-based tools and applications.

6. THE ER AUTHORING TOOL IN BRIEF

Let A and B two events. Let us also assume that when an event A or B happens, those events are characterized by a value, as well as a timestamp indicating the event start time and finish time.

The current version (v1.0) of the ER Authoring Tool supports the following temporal operators (constructs) over these events:

– (A OR B) ≡ (B OR A): The resulting event occurs when at least one of A, B occurs.

– (A AND B) ≡ (B AND A): The resulting event occurs when A and B occur concurrently.

– (A MINUS B): The resulting event occurs when A occurs and B doesn't. Similarly, (B MINUS A): The resulting event occurs when B occurs and A doesn't.

– (A XOR B) ≡ (B XOR A): The resulting event occurs when only one of the A, B event occur. In other words: (A XOR B) ≡ (B XOR A) ≡ (A MINUS B) OR (B MINUS A).

– DetectChange(A): The resulting event occurs when the value of event A has changed.

The results of these operators are demonstrated in Figure 2.

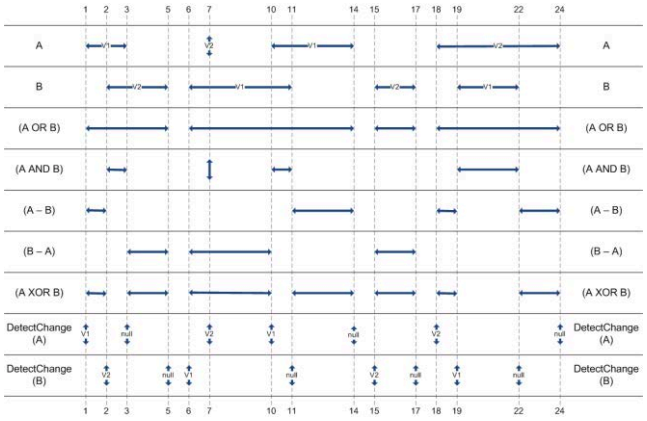

Figure 2. Use of Temporal Operators

The current version of the ER Authoring Tool also supports the following logical operators:

– Filter(A, LogicalExpression): Returns the instances of event A that meet some logical constraints

– LogAND(A, B, LogicalExpression): Returns an event if A and B exist under some conditions

All the above-mentioned operators are available for the definition of ER rules, via the tool's Graphical User Interface (Figure 3). The GUI consists of: (a) the design canvas where the user designs rules, (b) the design palette with the available operators, (c) the design outline, where the user may observe the entire ER design and (d) the property viewer where the user can define the properties (e.g. name, conditions) of an operator in use (Figure 4).

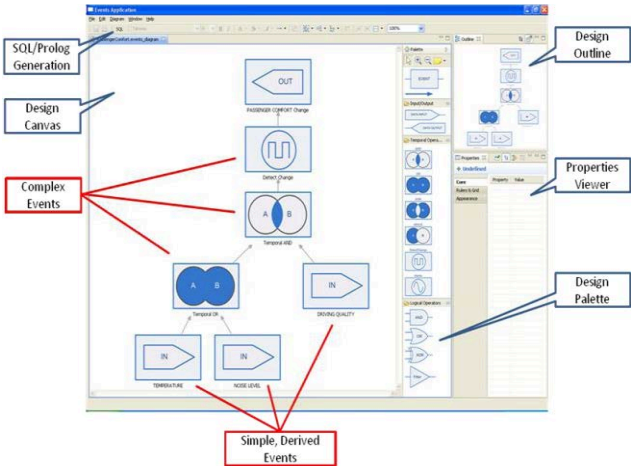

Figure 3. ER Authoring Tool GUI

In order to design a new rule, the user should select the desired operators and connections from the design palette and create a graph of events (from derived events to composite events). The user may also define properties (e.g. name, conditions, etc.) of these operators through the property editor. The next step is to check if the defined ER rule is valid according to the Validation Model. To do so, the user presses the "Validate" button from the menu. If no errors come up the rule is valid. Finally, if for example we want to generate a rule in SQL we have to press the "SQL Generation" button. The rule is then produced and ready to use after being compiled.

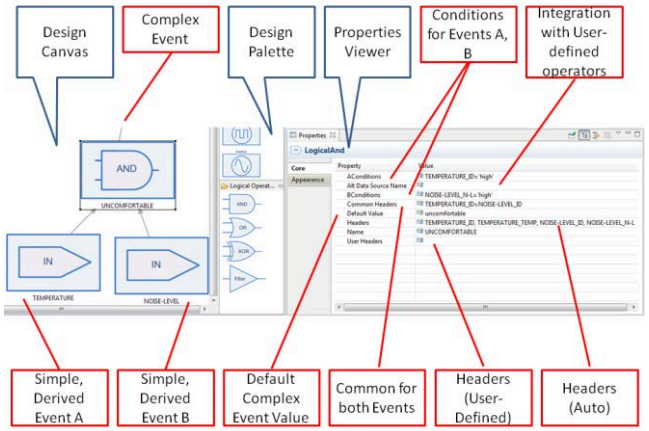

Figure 4. Properties Viewer for LogAnd

Below we present an example of an ER rule, and the corresponding output of the ER Authoring Tool.

Assume that the user of the tool wants to define the following event recognition rules:

```
Punctuality = "Punctual" when:
(StopEnter="Scheduled" OR "Early") AND
(StopLeave="Scheduled")

Punctuality = "nonPunctual" when:
(StopEnter = "Late") OR
(StopLeave="Early") OR
(StopLeave = "Late")
```

Version 1.0 of the ER Authoring tool supports the definition of SQL-based rules. More precisely, for the above example the following code that calls the related functions implementing the SQL queries will be generated:

```
//========= SYNTAX EXPLANATION

FILTER(INPUT, OUTPUT, [HEADERS], [CONDITIONS]);

LogicAND(IN1, IN2, OUT, [HEADERS], [CONDITIONS]);

//========= Punctuality Value = Punctual

FILTER("StopEnter","OUT1",
       [VehicleID, StopCode, Value],
       [Value="Early" or "Scheduled"]);

FILTER("StopLeave","OUT2",
       [VehicleID, StopCode, Value],
       [Value="Scheduled"]);

LogicAND("OUT1","OUT2","Punctuality",
       [VehicleID, StopCode, Value="Punctual"],
       [(OUT1.VehicleID = OUT2.VehicleID) AND
       (OUT1.StopCode = OUT2.StopCode]);

//========= Punctuality Value = NonPunctual

FILTER("StopEnter","Punctuality",
       [VehicleID,StopCode,Value="NonPunctual"],
       [Value="Late"]);

FILTER("StopLeave","Punctuality",
       [VehicleID,StopCode,Value="NonPunctual"],
       [Value="Early" or "Late"]);
```

Currently, we are in the process of extending the tool, to fully support the Event Calculus language syntax in prolog.

7. CONCLUSIONS

Currently there exist several tools for Complex Event Recognition, varying from design platforms for business process modeling (BPM) to advanced Complex Event Processing (CEP) engines. Several efforts have been reported in literature aiming to support domain experts in the process of defining event recognition (ER) rules. However, few of them offer graphical design environments for the definition of such rules, limiting the broad adoption of ER systems. In this paper, we present a graphical Event Definition Authoring Tool, referred to as the Event Recognition Designer Toolkit (ERDT) with which, a domain expert can easily design event recognition rules on temporal data and produce standalone Event Recognizers.

8. ACKNOWLEDGMENTS

This work has been supported by the EC-funded project PRONTO (http://www.ict-pronto.org/)

9. REFERENCES

[1] Aleri. 2010. http://www.aleri.com/. Visited Feb. 2011.

[2] Chaudhuri, S. and Dayal, U. 1997. An overview of data warehousing and olap technology. SIGMOD Rec. 26, 1, 65-74.

[3] IBM. 2008. Business event processing white paper, websphere software.

[4] Gualtieri, M. and Rymer, J. 2009. The Forrester WaveTM: Complex Event Processing (CEP) Platforms, Q3 2009.

[5] Oracle. 2010. http://www.oracle.com/technologies/soa/complex-event-processing.html. Visited Feb. 2011.

[6] Progress-Apama. 2010. http://www.ict-pronto.org/. Visited Apr. 2011.

[7] PRONTO. 2009. http://web.progress.com/it-need/complex-event-processing.html. Visited Feb. 2011.

[8] Streambase. 2010a. http://www.streambase.com/. Visited Feb. 2011.

[9] Streambase. 2010b. http://streambase.com/developers/docs/latest/streamsql/index.html. Visited Feb. 2011.

[10] Tibco. 2010. http://www.tibco.com/software/complex-event-processing/ businessevents/ default.jsp. Visited Feb. 2011.