ECAI 2012
Luc De Raedt et al. (Eds.)
© 2012 The Author(s).
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License.
doi:10.3233/978-1-61499-098-7-943

Event Processing for Intelligent Resource Management

Alexander Artikis[1] and **Robin Marterer**[2] and **Jens Pottebaum**[2] and **Georgios Paliouras**[1]

Abstract. The need for intelligent resource management (IRM) spans across a multitude of applications. To address this requirement, we present EP-IRM, an event processing system recognising composite events given multiple sources of information in order to support IRM. EP-IRM has been deployed in two real-world applications. Moreover, with a small effort it may be used in a wide range of applications requiring IRM. We present an evaluation of the system, and discuss the lessons learnt during its development and deployment.

1 Introduction

Organisations collect data in various formats, but they cannot fully utilise these data to support their resource management. It is evident that the analysis and interpretation of the collected data need to be automated, so that large data volumes can be transformed into operational knowledge. Events are particularly important pieces of knowledge, as they represent activities of special significance for and within an organisation. Therefore, the processing and, in particular, the recognition of events, is of utmost importance.

Systems for event recognition ('event pattern matching') accept as input time-stamped simple, derived events (SDE). A SDE ('low-level event', 'short-term activity') is the result of applying a computational derivation process to some other event, such as an event coming from a sensor [9]. Using SDE as input, event recognition systems identify composite events (CE) of interest—collections of events that satisfy some pattern. The 'definition' of a CE ('high-level event', 'long-term activity') imposes temporal and, possibly, atemporal constraints on its subevents ('members'), that is, SDE or other CE.

We present an event processing system recognising CE for intelligent resource management (IRM). The need for IRM spans across many applications. In emergency rescue operations, for example, there is a pressing need for real-time decision support that facilitates the fastest possible completion of the operation with the minimum possible casualties. An operation manager needs to be aware of a dynamically evolving emergency and decide, in real-time, how to deploy and manage a rescue team in order to complete a rescue operation. Additionally, there is a need for off-line retrospective analysis of operations for debriefing and training sessions.

In the proposed system, hereafter EP-IRM, data is constantly acquired, synchronised and aggregated from various types of sensor installed in the infrastructure of the end user (for example, fire brigade), and from various modes of interaction between the actors of the application in hand (for instance, fire brigade officers). The aggregated data is analysed and enhanced with spatial information in order to extract SDE. Then, event recognition techniques are applied on the SDE streams in order to recognise, in real-time, CE. Given a SDE stream concerning the interactions of rescue workers and climate sensor data, for instance, the criticality of a rescue operation is automatically detected for the benefit of the operation manager, responsible for resource management. A user-friendly IRM component provides an interface to EP-IRM which is used to support the decision-making process at that level.

EP-IRM, therefore, seamlessly integrates various types of novel event processing component for real-time CE recognition given multiple sources of information. EP-IRM has been deployed, in the context of the PRONTO project[3], in two very different application domains: management of emergency rescue operations and city transport. Furthermore, the evaluation of EP-IRM shows that it may support real-time decision-making in most application domains.

2 Demonstration Cases

EP-IRM has been used for supporting city transport management (CTM) in Helsinki, Finland. Buses and trams are equipped with in-vehicle units that send GPS coordinates, acceleration information, in-vehicle temperature and noise level to a central server providing information about the current status of the transport system (for example, the location of buses and trams on the city map). Given the SDE extracted from such sensors, and from other data sources such as digital maps, CE are recognised related to the punctuality of a vehicle, passenger and driver comfort, passenger and driver safety, and passenger satisfaction. The recognised CE are made available to the transport control centre in order to facilitate decision-making.

EP-IRM has also been used for supporting the emergency rescue operations (ERO) of the Fire Department of Dortmund, Germany. Input for CE recognition is gathered during regular daily business—using fire detection systems and weather information services—as well as in exceptional situations, that is, during an operation. An emergency and its evolution are observed by smoke and gas detectors. The emergency response is monitored by GPS, fuel and water sensors mounted on the vehicles used in the response. The SDE detectors operating on these sensors send data to control centres. Furthermore, rescue officers perform reconnaissance actions and communicate results to command posts—commanders enter information about the environment, the emergency and the response into support systems. The communication channel and the interaction with such systems are also used for SDE detection. The CE recognised on the SDE streams concern changes in the need for operations and the criticality of operations, among others—such CE allow decision-makers to perform goal-oriented improvisation and disposition of resources.

Both ERO and CTM require event processing in critical, complex situations that are characterized by the need for decisions, the existence of alternative options, high interdependencies between and intransparency of system elements, irreversible actions and challenging time constraints [3]. Decision-makers are part of high reliability organizations, managing assessable but unforeseeable risks. In the case of an abnormal situation, such as an emergency, it is necessary

[1] NCSR Demokritos, Greece, email: {a.artikis, paliourg}@iit.demokritos.gr
[2] Universität Paderborn, C.I.K., Germany, email: {marterer, pottebaum}@cik.uni-paderborn.de

[3] `http://www.ict-pronto.org/`

to recognise the events of interest, interpret the context, derive action alternatives and make a decision. Due to time constraints, it is often impossible to analyse the effects of each possible action. There is a need, therefore, for CE recognition and decision support [7].

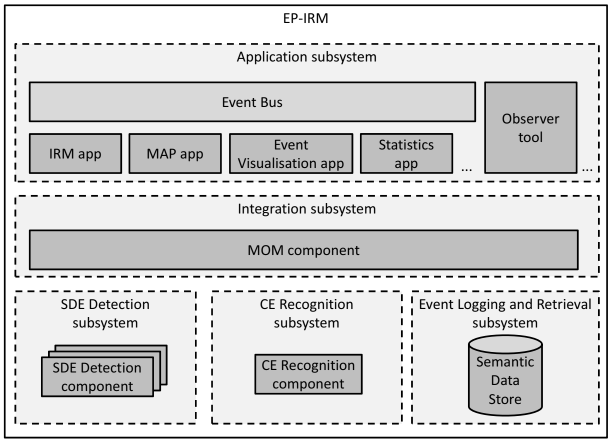


Figure 1. EP-IRM architecture.

3 System Architecture

EP-IRM is based upon the principles of event-driven, service-oriented architectures [10]. It is divided into subsystems containing one or more components—see Figure 1. Due to the modular design and loose coupling of EP-IRM, components written in various programming languages can be added, replaced or removed without much effort. All subsystems are connected through the Message-Oriented Middleware (MOM). The MOM is based on HornetQ, which is the JBoss application server implementation of the Java Message Service (JMS) standard. Messages in the MOM represent events communicated between subsystems.

EP-IRM includes components detecting SDE from audio, video, text, location, temperature, acceleration, and vehicle engine data. Following the publish-subscribe pattern, components may act as event producers or event consumers. For example, the SDE detection components consume raw events coming from sensors (microphones, cameras, GPS, etc) in order to produce SDE. The CE recognition component consumes SDE in order to produce CE. All events are logged in a semantic data store which is accessible via the MOM.

The application subsystem includes the web applications ('apps') directly available to the user through a web-based interface. The Statistics app, for example, calculates and visualises event-based statistical information, such as bus/tram fuel consumption in CTM. To facilitate the communication between apps, an event bus is employed. Apart from web applications, the application subsystem is open for integration with stand-alone software tools like 'Observer'. This tool supports post-operation use cases such as debriefings and operating reports. The EP-IRM application subsystem and the user interface are further discussed in Section 5.

The system is fully functional and integrated into the end-users' infrastructures in Helsinki and Dortmund. The two installations do not have exactly the same modules as in CTM and ERO there are different types of sensor and user interaction. Next, we briefly present the CE recognition component. Due to space limitations we cannot present in detail the remaining EP-IRM modules.

4 Composite Event Recognition

Our component for CE recognition is a logic programming (Prolog) implementation of an Event Calculus (EC) dialect. EC [8] is a logic programming language for representing and reasoning about events and their effects. The benefits of a logic programming approach to CE recognition are well-documented [11]: such an approach has a formal, declarative semantics, is highly expressive, has direct routes to machine learning for automatically constructing CE definitions (see, for instance, [6]), and has direct routes to reasoning under uncertainty for addressing the issues of noisy SDE streams and imprecise CE definitions (see, for example, [5]). The use of EC has additional advantages: the process of CE definition development is considerably facilitated, as EC includes built-in rules for complex temporal representation and reasoning, including the formalisation of inertia. With the of EC one may develop intuitive, succinct CE definitions, facilitating the interaction between CE definition developer and domain expert, and allowing for code maintenance.

4.1 Representation

For the EC dialect presented here, 'Event Calculus for Run-Time reasoning' (RTEC), the time model is linear and includes integers. Where F is a *fluent*—a property that may have different values at different time-points—the term $F = V$ denotes that fluent F has value V. Boolean fluents are a special case in which the possible values are true and false. Informally, $F = V$ holds at a particular time-point if $F = V$ has been *initiated* by an event at some earlier time-point, and not *terminated* by another event in the meantime (law of inertia).

Table 1. Main predicates of RTEC.

Predicate	Meaning
happensAt$(E,\ T)$	Event E is occurring at time T
initially$(F = V)$	The value of fluent F is V at time 0
holdsAt$(F = V,\ T)$	The value of fluent F is V at time T
holdsFor$(F = V,\ I)$	I is the list of maximal intervals for which $F = V$ holds continuously
initiatedAt$(F = V,\ T)$	At time T a period of time for which $F = V$ is initiated
union_all$(L,\ I)$	I is the list of maximal intervals produced by the union of the lists of maximal intervals of list L
intersect_all$(L,\ I)$	I is the list of maximal intervals produced by the intersection of the lists of maximal intervals of list L
relative_complement_all(I', L, I)	I is the list of maximal intervals produced by the relative complement of the list of maximal intervals I' with respect to every list of maximal intervals of list L

An *event description* in RTEC includes axioms that define the event occurrences (with the use of the happensAt predicates), the effects of events (with the use of the initiatedAt predicates), and the values of the fluents (with the use of the initially, holdsAt and holdsFor predicates). Table 1 summarises the RTEC predicates available to the CE definition developer. Variables, starting with an upper-case letter, are assumed to be universally quantified unless otherwise indicated. Predicates and constants start with a lower-case letter.

The city transport officials are interested in computing, for example, the intervals during which a vehicle is (non-)punctual. This may be achieved in RTEC as follows:

$$\text{initially}(punctuality(\_,\_) = punctual) \tag{1}$$

$$\begin{aligned}&\text{initiatedAt}(punctuality(Id,\ VT) = punctual,\ T) \leftarrow\\&\quad \text{happensAt}(enter\_stop(Id,\ VT, Stop, scheduled),\ \_),\\&\quad \text{happensAt}(leave\_stop(Id,\ VT, Stop, scheduled),\ T)\end{aligned} \tag{2}$$

$$\begin{aligned}&\text{initiatedAt}(punctuality(Id,\ VT) = punctual,\ T) \leftarrow\\&\quad \text{happensAt}(enter\_stop(Id,\ VT, Stop, early),\ \_),\\&\quad \text{happensAt}(leave\_stop(Id,\ VT, Stop, scheduled),\ T)\end{aligned} \tag{3}$$

$$\begin{aligned}&\mathsf{initiatedAt}(punctuality(Id,\ VT) = non\_punctual,\ T) \leftarrow\\ &\quad \mathsf{happensAt}(leave\_stop(Id,\ VT,\ \_,\ early),\ T)\end{aligned} \tag{4}$$

$$\begin{aligned}&\mathsf{initiatedAt}(punctuality(Id,\ VT) = non\_punctual,\ T) \leftarrow\\ &\quad \mathsf{happensAt}(leave\_stop(Id,\ VT,\ \_,\ late),\ T)\end{aligned} \tag{5}$$

$enter\_stop$ and $leave\_stop$ are instantaneous SDE, determined from sensor data and a database of timetable information. Id represents the id of a vehicle, VT represents the type of a vehicle (bus or tram), $Stop$ is the code of a stop, and '_' is an 'anonymous' Prolog variable. Initially, every vehicle is punctual. Thereafter $punctuality$ is affected by the $enter\_stop$ and $leave\_stop$ events. A vehicle is said to be punctual if it arrives at a stop on or before the scheduled time, and leaves the stop at the scheduled time. A vehicle is said to be non-punctual if it leaves the stop before or after the scheduled time. Computing the maximal intervals during which a vehicle is continuously (non-)punctual is achieved by computing the maximal intervals of $punctuality$ using the built-in holdsFor predicate.

Transport officials are also interested in recognising punctuality *change*. Consider the following formalisation:

$$\begin{aligned}&\mathsf{happensAt}(punctuality\_change(Id,\ VT,\ Value),\ T) \leftarrow\\ &\quad \mathsf{holdsFor}(punctuality(Id,\ VT) = Value,\ I),\\ &\quad (T, \_) \in I,\ T \neq 0\end{aligned} \tag{6}$$

This rule uses holdsFor to compute the maximal intervals for which a vehicle is continuously (non-)punctual. Punctuality changes at the first time-point of each of these intervals—see the penultimate condition of rule (6).

Briefly, to compute the maximal intervals during which a fluent F has value V continuously, that is, to compute holdsFor$(F = V, I)$, we find all time-points T_s in which $F = V$ is initiated, and then, for each T_s, we compute the first time-point T_f after T_s in which $F = V$ is terminated. The time-points in which $F = V$ is initiated are computed with the use of initiatedAt$(F = V, T)$ rules. The time-points in which $F = V$ is terminated are computed with the use of initiatedAt$(F = V', T)$ rules where $V \neq V'$.

In addition to the domain-independent definition of holdsFor, an event description may include domain-dependent holdsFor rules. Such rules use interval manipulation constructs. RTEC supports three such constructs: union_all, intersect_all and relative_complement_all (see Table 1). I in union_all$(L,\ I)$ is a list of maximal intervals that includes each time-point of each list of L. I in intersect_all$(L,\ I)$ is a list of maximal intervals that includes each time-point that is part of all lists of L. I in relative_complement_all$(I',\ L,\ I)$ is a list of maximal intervals that includes each time-point of I' that is not part of any list of L. Three example domain-dependent holdsFor rules are the following:

$$\begin{aligned}&\mathsf{holdsFor}(driving\_quality(Id,\ VT) = high,\ I) \leftarrow\\ &\quad \mathsf{holdsFor}(driving\_style(Id,\ VT) = uncomfortable, I'),\\ &\quad \mathsf{holdsFor}(driving\_style(Id,\ VT) = unsafe,\ I''),\\ &\quad \mathsf{holdsFor}(punctuality(Id,\ VT) = punctual,\ I'''),\\ &\quad \mathsf{relative\_complement\_all}(I''',\ [I', I''],\ I)\end{aligned} \tag{7}$$

$$\begin{aligned}&\mathsf{holdsFor}(driving\_quality(Id,\ VT) = medium,\ I) \leftarrow\\ &\quad \mathsf{holdsFor}(driving\_style(Id,\ VT) = uncomfortable, I'),\\ &\quad \mathsf{holdsFor}(punctuality(Id,\ VT) = punctual,\ I''),\\ &\quad \mathsf{intersect\_all}([I', I''],\ I)\end{aligned} \tag{8}$$

$$\begin{aligned}&\mathsf{holdsFor}(driving\_quality(Id,\ VT) = low,\ I) \leftarrow\\ &\quad \mathsf{holdsFor}(driving\_style(Id,\ VT) = unsafe,\ I'),\\ &\quad \mathsf{holdsFor}(punctuality(Id,\ VT) = non\_punctual,\ I''),\\ &\quad \mathsf{union\_all}([I', I''],\ I)\end{aligned} \tag{9}$$

$punctuality$ was defined by rules (1)–(5), while the definition of $driving\_style$ is omitted to save space. High quality driving is recognised when a vehicle is punctual and the driving style is neither unsafe nor uncomfortable. Medium quality driving is recognised when the driving style is uncomfortable and the vehicle is punctual. Low quality driving is recognised when the driving style is unsafe or the vehicle is non-punctual.

The use of interval manipulation constructs leads to a simple representation of the CE concerning driving quality. In the absence of these constructs, one would have to adopt the traditional style of EC representation, that is, identify all conditions in which $driving\_quality(Id,\ VT) = high$ (respectively $= medium$, $= low$) is initiated, all conditions in which this CE is terminated, and then use the domain-independent holdsFor predicate to compute the maximal intervals of the CE. Such a formalisation would be more complex than the representation of rule (7) (respectively, rules (8) and (9)). In general, the use of RTEC constructs manipulating intervals—union_all, intersect_all and relative_complement_all—may significantly simplify the definitions of durative CE. With the use of union_all, for example, we are able to develop succinct representations of most definitions of the durative CTM CE. The interval manipulation constructs can also lead to much more efficient CE recognition.

4.2 Reasoning

Typically, CE recognition has to be efficient enough to support real-time decision-making, and scale to very large numbers of SDE. These SDE may not necessarily arrive at the CE recognition system in a timely manner, that is, there may be a (variable) delay between the time at which SDE take place and the time at which they arrive at the CE recognition system. Moreover, SDE may be revised, or even completely discarded in the future. Consider, for example, the case where the parameters of a SDE were originally computed erroneously and are subsequently revised, or the retraction of a SDE that was reported by mistake, and the mistake was realised later [1]. Note that SDE revision is not performed by the CE recognition system, but by the underlying SDE detection system. The effects of SDE revision are computed by the CE recognition system, provided that the latter supports such functionality.

RTEC performs run-time CE recognition by computing and storing the maximal intervals of fluents and the time-points in which events occur. CE recognition takes place at specified query times $Q_1, Q_2, \ldots$. At each query time Q_i only the SDE that fall within a specified interval—the 'working memory' or 'window' (*WM*)—are taken into consideration: all SDE that took place before or on $Q_i - WM$ are discarded. This is to make the cost of CE recognition dependent only on the size of *WM* and not on the complete SDE history. The size of *WM*, as well as the temporal distance between two consecutive query times—the 'step' ($Q_i - Q_{i-1}$)—is chosen by the user. Consider the following cases:

- $WM < Q_i - Q_{i-1}$. In this case, the effects of the SDE that took place in $(Q_{i-1}, Q_i - WM]$ will be lost.
- $WM = Q_i - Q_{i-1}$. In this case, no information will be lost, *provided that* all SDE arrive at RTEC in a timely manner, and there is no SDE revision. If SDE do not arrive in a timely manner, then the effects of SDE that took place before Q_i but arrived after Q_i will be lost. Furthermore, if SDE are revised, the effects of the revision of SDE that took place before Q_i and were revised after Q_i will be lost.
- $WM > Q_i - Q_{i-1}$. In the common case that SDE arrive at RTEC with delays, or there is SDE revision, it is preferable to make *WM* longer than the step. In this way, it will be possible to compute,

at Q_i, the effects of SDE that took place in $(Q_i - WM, Q_{i-1}]$, but arrived at RTEC after Q_{i-1}. Moreover, it will be possible to compute, at Q_i, the effects of the revision of SDE that took place in $(Q_i - WM, Q_{i-1}]$ and were revised after Q_{i-1}.

Even when $WM > Q_i - Q_{i-1}$ information may be lost. The effects of SDE that took place before or on $Q_i - WM$ and arrived after Q_{i-1} are lost. Similarly, the effects of the revision of SDE that took place before or on $Q_i - WM$ and were revised after Q_{i-1} are lost. To reduce the possibility of losing information, one may increase the size of *WM*; in this case, however, recognition efficiency will decrease.

RTEC is the most appropriate EC dialect for run-time CE recognition as, among others, it is the only EC dialect operating on WM, being therefore independent of the complete SDE history. A detailed account of our CE recognition component and a comparison with related (EC-based) approaches are given in [2].

When SDE arrive with a variable delay, or when SDE are revised by the SDE detection components, some of the CE intervals computed and stored at an earlier query time may be, partly or completely, retracted at the current or a future query time. Depending on the requirements of the application under consideration, RTEC may report to the user:

- CE as soon as they are recognised, even though the intervals of these CE may be partly or completely retracted in the future.
- CE whose intervals may be partly, but not completely, retracted in the future.
- CE whose intervals will not be, even partly, retracted in the future.

5 Evaluation

By far the most computationally expensive EP-IRM component is the CE recognition component. Moreover, CTM proved to be more computationally demanding than ERO with respect to CE recognition. Therefore, we present experimental results concerning CE recognition for CTM. The experiments were performed on a computer with Intel i7 950@3.07GHz processors and 12GiB RAM, running Ubuntu Linux 11.04 and YAP Prolog 6.2.0.

Figure 2 shows the results of experiments concerning CE recognition at rush hour in Helsinki. At most 1050 vehicles, that is, 80% of the total number of available vehicles, operate at the same time in Helsinki during rush hour. Due to the unavailability of real datasets at that scale, we simulated rush hour operations using synthetic datasets. Experts estimate that no more than 350 SDE can be detected per second on the 1050 operating vehicles. We were thus able to test RTEC under the maximum expected frequency of SDE.

Figure 2 presents the recognition times of RTEC in CPU milliseconds (ms) concerning three sets of experiments. First, we used a single processor to perform CE recognition for all 1050 vehicles. In this case, the intervals of 21000 CE (1050 vehicles $\times$ 20 CE per vehicle) are computed and stored. Second, we used four processors in parallel. Each instance of RTEC running on a processor performed CE recognition for one quarter of all operating vehicles, that is, 263 vehicles, computing and storing the intervals of 5260 CE. Third, we used all eight processors of the computer in parallel. Each instance of RTEC running on a processor performed CE recognition for one eighth of all operating vehicles, that is, 132 vehicles, and computed and stored the intervals of 2640 CE.

In all sets of experiments the input was the same: SDE coming from *all* 1050 vehicles. In other words, there was no filtering of SDE in these experiments to restrict the input relevant for each processor.

The datasets used for evaluation include SDE that are not chronologically ordered. The step is set to 1 sec (350 SDE), while *WM*

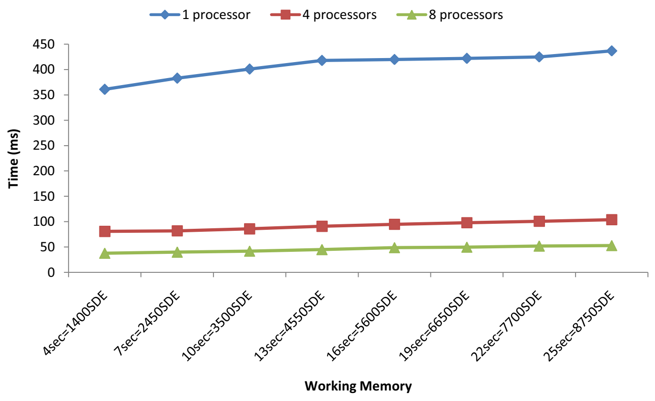


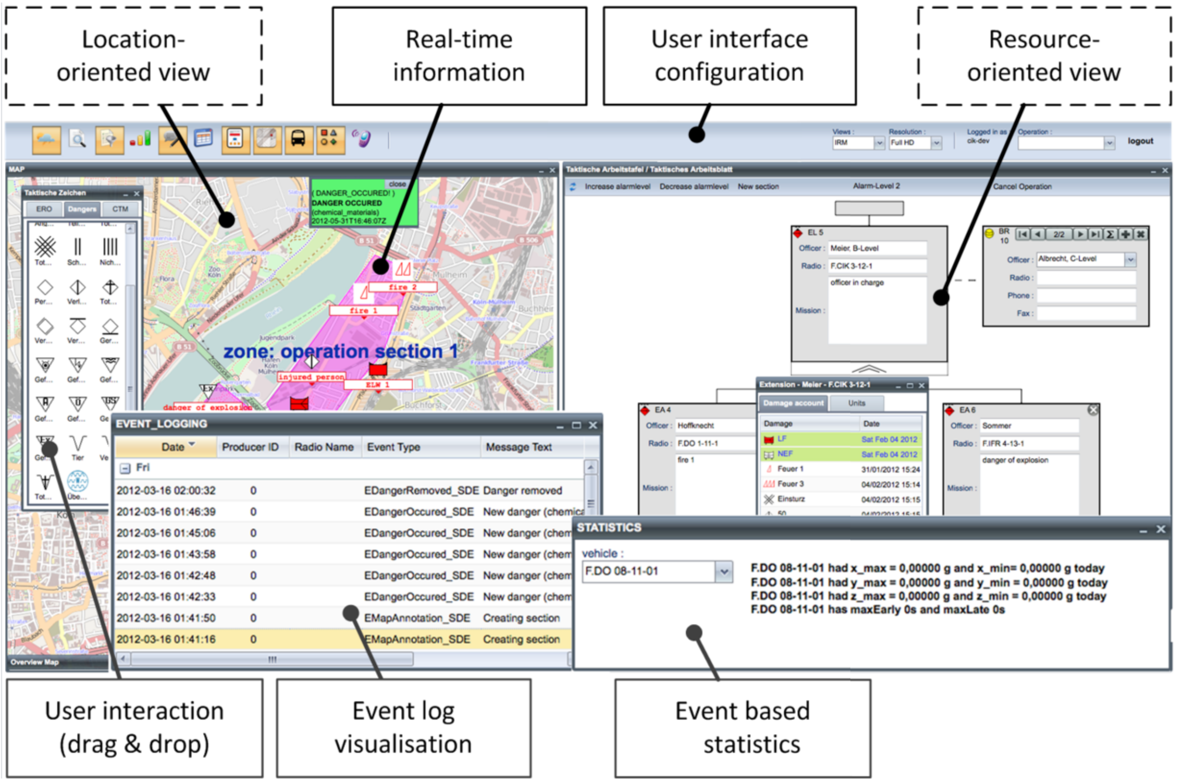
Figure 2. Total RTEC time: CE recognition during rush hour in Helsinki, step set to 1 sec = 350 SDE.

ranges from 4 sec (1400 SDE) to 25 sec (8750 SDE). We found (in experiments not presented here due to lack of space) that reducing the step size reduces recognition times very slightly. Given the current infrastructure in Helsinki, a 10 sec *WM* is sufficient, that is, a delay in the arrival of a SDE is expected to be less than 10 sec. Other CTM infrastructures may require different *WM* sizes.

Figure 2 shows that we can achieve a significant performance gain by running RTEC in parallel on different processors. Such a gain is achieved without requiring SDE filtering.

Apart from quantitative evaluation, we performed qualitative, user-oriented evaluation—we estimated the impact of EP-IRM on the end user organisations by means of interviews. Qualitative evaluation is related to questions of effectiveness (does EP-IRM fit to its intended purpose?), efficiency (does EP-IRM facilitate quick task conduction?) and user satisfaction (do users feel comfortable using EP-IRM?). In what follows, we briefly discuss the qualitative evaluation of EP-IRM on ERO. This type of evaluation was considerably aided by the visualisation capabilities of EP-IRM—see Figure 3. Use cases are implemented by several apps which build the integrated user interface. Each of the apps is represented by a window. Real-time information, including the CE and SDE recognised at each time, is continuously updated without user interaction (push paradigm). A user bar allows the configuration of apps and views. On the left hand side, each app can be switched on or off by a button. On the right hand side, different views for different user roles and operation context can be selected.

The IRM app, for example, shows a logical view of the system's status. For example, in ERO the IRM app displays a tree view of the rescue operation command structure current at each time. Moreover, it shows the list of dangers of an operation, highlighting new ones in order to enable a commander to react to them. The MAP app presents a geo-based view of the system's status. For example, it displays positions of vehicles (fire brigade vehicles in ERO and buses and trams in CTM) and additional vehicle information or marked zones. Interactions between vehicles and spatial entities (such as an emergency area in ERO or a a dangerous intersection in CTM) are prominently highlighted. All apps are connected to each other (via an event bus). For instance, when a new danger occurs and is shown in the IRM app, the user can click on it and its position as well as related information (such as a photo of the danger location) will be presented on the MAP app. Apps act as event consumers—for instance, the Event Visualisation app consumes SDE and CE in order to display them in



EVENT_LOGGING

Date	Producer ID	Radio Name	Event Type	Message Text
Fri				
2012-03-16 02:00:32	0		EDangerRemoved_SDE	Danger removed
2012-03-16 01:46:39	0		EDangerOccured_SDE	New danger (chemica
2012-03-16 01:45:06	0		EDangerOccured_SDE	New danger (chem
2012-03-16 01:43:58	0		EDangerOccured_SDE	New danger (chem
2012-03-16 01:42:48	0		EDangerOccured_SDE	New danger (chem
2012-03-16 01:42:33	0		EDangerOccured_SDE	New danger (chem
2012-03-16 01:41:50	0		EMapAnnotation_SDE	Creating section
2012-03-16 01:41:16	0		EMapAnnotation_SDE	Creating section

STATISTICS

vehicle : F.DO 08-11-01

F.DO 08-11-01 had x_max = 0,00000 g and x_min= 0,00000 g today
F.DO 08-11-01 had y_max = 0,00000 g and y_min = 0,00000 g today
F.DO 08-11-01 had z_max = 0,00000 g and z_min = 0,00000 g today
F.DO 08-11-01 has maxEarly 0s and maxLate 0s

Figure 3. The EP-IRM user interface for the real-time use cases.

a time-line to the user. Moreover, apps act as event producers. Consider, for example, the case in which a rescue operation commander creates an operational section using the IRM app in order to better manage the operation. In this case the event 'section created' will be produced and published in the MOM so that other components, such as the CE recognition component, may consume it. Similarly, when a commander drags & drops tactical symbols onto the map to denote a danger, the 'danger occurred' event is transmitted to the MOM.

The main goal of the qualitative evaluation is to estimate the user satisfaction. We correlate user satisfaction with the added value perceived by practitioners when using EP-IRM. All fire officers answered questions targeting the added value offered by EP-IRM. For real-time decision support, as well as debriefing and training sessions, users assessed the innovation with respect to known information management solutions. All interviewees acknowledged the potential of CE recognition in large-scale operations. These operations are characterised by a high number of influencing elements (for example, danger events, resources performing actions and communication) with complex relations. Interviewees stated that the recognised CE add significant value to domain-specific, state-of-the-art information management tools. Not surprisingly, higher-level officers saw increased added value than lower-level officers.

6 Lessons Learned

One of the challenges we had to face during the project concerns the understanding and use of the term 'event'. As an example, nearly 80% of ERO interviewees reduced the scope of 'event' to an emergency (for example, dangers happening in a specific environment)—they did not correlate this term with the emergency *response* (for instance, forces starting fire fighting) which is prominent in the ERO CE definitions. We found it impossible to introduce a shared understanding of 'event', and accepted the terminology divergence in our interviews with end users concerning requirements specification, including CE definition, as well as system evaluation. The researchers conducting the interviews had to ensure that statements made by end users were appropriately interpreted.

A more significant challenge we had to face concerns CE definition. Although end users had initially some idea about the CE of interest, the definitions of these CE were unclear. In other words, all conditions in which a CE should be recognised were not clear. Consequently, CE definitions were frequently updated during the lifetime of the project, sometimes as a result of new sensor types that became available at the end user infrastructure—SDE detected on the new sensor types lead to more accurate CE recognition. The use of RTEC facilitated considerably the interaction between CE definition developers (programmers) and end users. With the use of RTEC we could produce succinct, intuitive CE definitions that could be understood, and sometimes directly manipulated, by end users.

The biggest challenge we had to face concerns data collection and annotation. Most CE represent 'abnormal' situations, such as emergencies in ERO, that rarely take place. Consequently, the collection of sufficient amounts of data including all anticipated CE required a lot of time. Moreover, given the fact that end users were unclear about the definitions of CE, data annotation (to provide the ground truth for CE recognition) by this group of stakeholders was very challenging. Data annotation was also challenging due to the fact that CE have relatively (very) short duration. To address these issues, we had to carefully plan data collection (for instance, align with the training exercise schedule of fire brigade officers). Moreover, we collected data at many stages of the development of EP-IRM, in order to allow for the refinement of this process—for example, give recommendations to members of the end user organisations on how to improve data collection and annotation. To allow for testing EP-IRM at the early stages of the project where sufficient data were unavailable, we developed data generators simulating CTM and ERO operations.

End users are not always able to quantitatively estimate event recognition. Therefore we had to extend the validation approach of [12] by allowing for qualitative evaluation. Users do not always think of milliseconds—they sometimes think about the processes before which a CE should be recognised. For example, the 'demand for additional resources' ERO CE should be recognised anytime before the following reconnaissance. One aspect that we did not anticipate concerns the fact that end users do not always have high performance requirements. For example, some rescue officers accepted delays up to one minute concerning the recognition of some CE because they would not be able to use this information (recognised CE) earlier. This is supported by the fact that briefings, debriefings and operating reports are highlighted as major use cases for EP-IRM in addition to real-time event recognition and visualisation.

Concerning recognition accuracy, the assumption of perfect precision and recall was challenged by end users. A 95% accuracy is acceptable by most users. The accuracy of EP-IRM, therefore, was found acceptable. The interviews showed that the impact of false negatives is diverse. In some cases, such as when a 'resource departed' in order to participate in an emergency operation, but this CE was not recognised, false negatives lead to an overestimation of the necessary resources, but have no negative influence on an emergency. The impact of false positives is much more critical up to 'not acceptable'. Only a few officers deviate from this judgement who would double-check information provided by an event processing system.

End users benefit from, and often demand, explanation facilities from the event processing system. When various recognised CE were presented to the users, an explanation concerning CE recognition was required ('drill down')—what are the occurrences of the subevents of the CE that lead to the CE recognition? Such a feature is deemed necessary both for run-time and off-line use of the system. Building upon Prolog's tracing facility, we can already offer a form of explanation facility.

7 Summary & Transferability

We presented EP-IRM, an event processing system supporting intelligent resource management. EP-IRM seamlessly integrates various types of novel event processing component for CE recognition given multiple sources of information, including various types of sensor and modes of actor interaction. The complex CTM and ERO CE definitions enabled us to perform a realistic evaluation of the performance of EP-IRM. According to the results of the use case survey of the Event Processing Technical Society [4], in most application domains there are at most 1000 SDE per second. Our experimental evaluation showed that EP-IRM supports real-time decision-making in such domains.

EP-IRM has been deployed in the two very different application domains of ERO and CTM. Below we discuss what needs to be done in order to use EP-IRM in other domains.

If necessary, the MOM may be replaced by any JMS implementation. Many apps are generic and may be used in several application domains (for example, the same MAP and Event Visualisation apps are used both in CTM and ERO), while some are domain-specific. The apps are independent and may be replaced, and new ones added, seamlessly. Most SDE detection components may be easily reused in any application domain. The audio SDE detection component, for example, consists of a model-free approach that may work in many fields without much adaptation. The video component for unusual SDE detection may be employed in any domain in which there is a need for monitoring (large) human crowds. Its sensitivity may be adapted by adjusting only a small number of parameters. Moreover, it is nearly independent from the camera viewpoint.

On the contrary, the speech detection component used in ERO has to be adapted heavily for each new application domain. Currently it is optimised for TETRA radio messages in German in the setting of fire fighter operations. Changing these conditions needs major effort.

In some application domains it may be required to use a subset of the EP-IRM modules—for example, there is no need for speech detection in CTM. The modular design and loose coupling of EP-IRM facilitates the process of removing/adding modules.

The reasoning algorithms of the CE recognition component are generic and may de directly used in any application domain. If EP-IRM is used for CTM or ERO using another transport or fire brigade infrastructure, for example CTM in London, then the CE definition library may need to be updated in order to meet the requirements of the new infrastructure—consider, for instance, the use of different SDE. In this case, transfer learning techniques may be used to port the existing CE definition library to the new domain. In any case, the techniques for incremental, supervised machine learning developed in the context of the project [6] may be used for the automatic construction/refinement of CE definitions. These techniques use SDE streams annotated with CE to continuously update the structure of existing CE definitions or construct definitions of new CE. Apart from members of the end user organisations, the annotation of SDE streams with CE may be performed by the users of the application under consideration—for instance, people using public transportation communicating when a particular vehicle is driven in an unsafe manner, when passenger satisfaction is reducing, and so on.

ACKNOWLEDGEMENTS

This work has been funded by the EU PRONTO project (FP7-ICT 231738).

REFERENCES

[1] D. Anicic, S. Rudolph, P. Fodor, and N. Stojanovic, 'Retractable complex event processing and stream reasoning', in *RuleML Europe*, pp. 122–137, (2011).

[2] A. Artikis, M. Sergot, and G. Paliouras, 'Run-time composite event recognition', in *Proceedings of DEBS*. ACM, (2012).

[3] A. Bennet and D. Bennet, *Handbook on Decision Support Systems*, chapter The Decision-Making Process for Complex Situations in a Complex Environment, 3–20, Springer, 2008.

[4] P. Bizzaro. Results of the survey on event processing use cases. Event Processing Technical Society, 2011.

[5] J. Filippou, A. Artikis, A. Skarlatidis, and G. Paliouras, 'A probabilistic logic programming event calculus', Technical report, Cornell University Library, (2012). `http://arxiv.org/abs/1204.1851v1`.

[6] N. Katzouris, J. Filippou, A. Skarlatidis, A. Artikis, and G. Paliouras. Final version of algorithms for learning event definitions. Deliverable 4.3.2 of PRONTO, 2012. Available from the authors.

[7] Gary A. Klein, 'A recognition-primed decision (RPD) model of rapid decision making', in *Decision Making in Action: Models and Methods*, 138–147, Norwood: Ablex Publishing Corporation, (1993).

[8] R. Kowalski and M. Sergot, 'A logic-based calculus of events', *New Generation Computing*, **4**(1), 67–96, (1986).

[9] D. Luckham and R. Schulte. Event processing glossary. Event Processing Technical Society, 2008.

[10] G. Mühl, L. Fiege, and P. R. Pietzuch, *Distributed event-based systems*, Springer, 2006.

[11] A. Paschke and M. Bichler, 'Knowledge representation concepts for automated SLA management', *Decision Support Systems*, **46**(1), 187–205, (2008).

[12] E. Rabinovich, O. Etzion, S. Archushin, and S. Ruah, 'Analyzing the behavior of event processing applications', in *DEBS*, ACM, (2010).