# Identification of critical situations via Event Processing and Event Trust Analysis

# Identification of Critical Situations via Event Processing and Event Trust Analysis

Massimiliano L. Itria[1], Melinda Kocsis-Magyar[2], Andrea Ceccarelli[3,5], Paolo Lollini[3,5], Gabriele Giunta[4], Andrea Bondavalli[3,5]

[1]Resiltech Srl, Pontedera, Italy
[2]Quanopt kft, Budapest, Hungary
[3]Univ. of Florence - Mathematics and Computer Science Dept., Florence, Italy
[4]Engineering Ingegneria Informatica S.p.A., Palermo, Italy
[5]CINI-Consorzio Interuniversitario Nazionale per l'Informatica, Florence, Italy

Received: Dec 03, 2015
Revised: Aug 15, 2016
Accepted: Oct 29, 2016

**Abstract**— In crisis management systems, situational awareness is usually at the basis of guiding the intervention process, and it is required to rapidly process data acquired from information sources on the field such as sensors or even humans. Given the variety and heterogeneity of sources and the amount of information that can be collected, together with the urgency of taking decisions, such information needs to be rapidly collected, filtered and aggregated in a form that can be used in subsequent machine-assisted decision support processes. At the same time, uncertainties in the input data or approximations in the processing phase may lead to an incorrect interpretation of the real situation in progress, which may generate mismanagements and severe consequences. This paper presents an event processor for crisis management systems that combines heterogeneous input sources to detect a critical situation. Complex event processing technology is applied for correlating data and creating events that describe the critical situation. Anomaly detection techniques are then used to analyze such events and detect possible anomalies, i.e. events not pertaining to the identified critical situation. The devised event processor creates trusted events that describe a critical situation merging inputs from heterogeneous and potentially untrusted sources. A prototype of the solution has been implemented and exercised within the crisis management system developed during the Secure! project. The experimental validation activities performed make use of different input sources, such as Twitter and sensors deployed on field (a doppler radar for people detection and accelerometers for vibrations detection). The objective of the experimental campaign is to show i) the adequacy of the solution to rapidly process the information and describe the critical situation, and ii) its capability in detecting anomalous events that could impair the accuracy of the description of the critical situation.

**Keywords**—Crisis Management System, Complex Event Processing, Event Correlator, Anomaly Detection, Decision Support System, Event Trust Analysis

## 1 Introduction

Crisis management systems aim to detect, recover or prevent critical situations. Depending on the goals of the system, such critical situations may span from natural crises as environmental phenomena, to technological crises caused by the human application of science and technology, to criminal or terrorist attacks crises [33], [36].

The modern experimentation in the context of crisis management systems takes into account the crowd-sourcing and crowd-sensing technologies: crowd-sourcing is the process of getting information online, from a crowd of people [1], while crowd-sensing is the involvement of a large, diffuse group of participants in the task of retrieving reliable data from a specific field [2]. Both technologies are means for capturing events from the real world in near real-time, exploiting the proliferation of modern mobile devices such as smart-phones and tablets, which can complement the information from response teams and from sensors deployed on the field.

As example, after the earthquake that struck Port-au-Prince in January 2010 [35] [7], a live crisis map of Haiti was launched using the Ushahidi platform. Information on the impact of the disaster was initially collected from online sources, including social media channels like Facebook and Twitter, and alerts received by SMS sent from citizens that wanted to signal their most urgent needs and location. Information coming from all sources was geo-located to build a crisis map; ten days after the earthquake, the Head of the US Federal Emergency Management Association (FEMA) recognized the map as the most comprehensive and up to date map available to the humanitarian community [13].

The ultimate objective of what is usually called Real-Time Situational Awareness (RTSA [3]) is to timely recognize, or even predict, *critical situations*, thus taking decisions to face them properly. For doing this it is paramount to have some enabling technologies that allow to retrieve and integrate information coming from heterogeneous sources, like mobile devices, social media and deployed

sensors (e.g., surveillance cameras), and to quickly process this large amount of data for *detecting and monitoring critical situations* and for providing information that supports the response process.

The correct detection of a critical situation strictly depends on the accuracy of the input data. Data collected from crowd may contain errors introduced by humans, either intentionally or unintentionally. Errors could also affect data measured by low quality sensors as for example surveillance cameras. In addition, errors may be introduced during the information processing due to the inherent coverage limitations of the adopted event detection algorithms. As a consequence, uncertainties in the input data or approximations in the processing phase may lead to an incorrect interpretation of the real situation in progress [28], and ultimately to mismanagements and severe consequences.

For these reasons, RTSA systems require mechanisms able to detect or even correct, if possible, wrong and unreliable information. If no correction is possible, such inaccurate data should be discarded so to have a more accurate detection of the emergency situations.

In this paper we present and assess an *event processor* for crisis management systems that i) allows detecting and describing a critical situation using data taken from heterogeneous sources, ii) is sufficiently flexible and generic to be applied to novel sources or crisis scenarios, iii) is based on a novel public ontology and event model described in standard languages (respectively, the Web Ontology Language OWL and the Unified Modeling Language UML), which can be further reused and modified also outside the event processor, iv) allows detecting potentially incorrect information that may negatively affect the accuracy of the critical situation detection process. The event processor is based on three pillars: i) an *ontology* and *event model*, which allows the classification and the integration of information coming from heterogeneous sources; ii) a set of *detection rules* implemented in a Complex Event Processor (CEP, [8]), which allows an efficient management of the pattern detection process for the considered input data stream; iii) a set of *anomaly detection algorithms*, to analyze the events produced by the correlator and detect possible anomalous events, i.e. events not pertaining to the identified critical situation (anomaly detection refers to the problem of finding patterns in data that do not conform to the expected behavior [31]). The final objective of the devised event processor is to create trusted events that describe a critical situation, merging inputs from heterogeneous and potentially unreliable sources.

The proposed solution has been implemented and integrated in a crisis management system developed in the context of the Secure! project [6]. Experimental results show the high efficiency of the event processor to handle a huge amount of data and to timely detect the critical situation in progress.

A preliminary version of the architecture of the event processor was presented in the informal paper [5]. Such paper contains: i) a preliminary, short description of the event processor, not including the ontology, the event model and the anomaly detection; ii) very limited validation activities not reported in this paper.

The rest of the paper is organized as follows. Section 2 provides basic definitions, and Section 3 describes the event model and the information fusion process that is adopted in the event processor. Section 4 describes the logical architecture and the functionalities of the event processor, while Section 5 describes the application of the event processor in a crisis management system. Section 6 shows the results of the performed validation activities, Section 7 discusses related works and finally conclusions are drawn in Section 8.

## 2 Basic terminology

For the sake of clarity we introduce in this section the terminology used throughout the paper.

The term *critical situation* has different meanings depending on the context or application domain it refers to. In this paper, a critical situation is defined as the occurrence of one or more *events* that might require a reaction.

As defined in [8], an *event* is "an occurrence within a particular system or domain; it is something that has happened, or is contemplated as having happened in that domain". This definition places the event concept into two different contexts: i) the real world in which events happen and ii) the realm of computerized event processing, where the word event is used to mean a programming entity that represents this occurrence. In this work, we distinguish between *events* that happen in the real world, and computerized *micro-events* and *macro-events*.

The *micro-events* concept belongs to an event ontology and represents simple real events involving one category from such ontology. Sample categories are people detection, face and logo recognition, recognition of abrupt sounds as explosions, guns firing, or detection of objects as knifes, guns, shotguns, hummers. Hence, the information that constitutes a *micro-event* is the textual description of the generic event, the time when it happened, its location, the category involved and the source that generated it. Sample sources are social media (e.g., Twitter), surveillance camera, proximity sensors for suspicious people movements detection and vibration sensors for detecting events such as explosions or earthquakes. In practice, *micro-events* are usually generated at the end of the features and information extraction process, leveraging on the acquired data processing (e.g., image, video and audio analysis, text mining, social network analysis) from the considered sources.

*Macro-events* are the aggregation result of the information contained in a set of micro-events which are correlated by spatial, temporal and causal relations. A macro-event contains more detailed and complete information than the related micro-events; it aims to describe a *critical situation* at a certain time and in a certain place. For example, it can describe the status of a demonstration in a part of the city.

When a critical situation happens, a number of micro-events are generated from the available heterogeneous sources; then, these data can be processed and aggregated to produce one or more macro-events, which can describe

the critical situation through time. On the basis of the information contained in the macro-events, response and recovery actions can be efficiently and effectively planned and actuated.

For example, let us suppose that a brawl between soccer supporters is happening; the presence of weapons and anomalous people behavior, which could be detected by sensors or by humans reporting information, are identified as a set of *micro-events*. These *micro-events* can be correlated in one or more *macro-events* that describe the current status of the brawl, and the macro-events can be used in a crisis management system to support response, e.g., to direct police towards the most dangerous supporters.

Given a set of macro-events referring to the same situation, we define a macro-event as *anomalous* if at least one of its characteristics/attributes is not consistent with the characteristics/attributes of the other macro-events. For example, an anomalous event can have different location or can happen at a different time w.r.t. to the majority of the other macro-events. Non anomalous macro-events are called *normal* macro-events.

## 3 From Micro-Events to Macro-Events

This section describes the events ontology, the event model, and the information fusion process that is built using such model. The event processor reported in Section 4 implements such ontology, model and process.

### 3.1 The Event Model

We rely on an ontology to associate one concept to each micro-event or macro-event. Such classification is a fundamental step in order to define correlations between events and consequently detect critical situations. We first discuss the ontology in use, and then the event model that exploits it. In this paper we define an ontology that is adequate for widely different events that may be of interest for a generic crisis management system.

Our ontology is defined in the Web Ontology Language (OWL); it was defined browsing event categories from DBPedia [44], a crowd-sourced community effort to extract structured information from Wikipedia and Wikidata and make this information available on the Web. The event categories have been selected taking into account the main disaster/crisis risk typologies, natural and man-made, to be faced. The ontology includes 8 event categories at the highest level (natural disaster, damage, terrorist attack, anomaly, theft, accident, violence, and entity recognition). The ontology is then modeled according to two further sub-categories, at the second and third level, respectively of 37 and 8 elements. Finally, at least one Wikipedia item or category was associated to each ontology event category, to support classification of the events generated from the text mining and processing of the integrated Web and Social sources. The entire ontology is freely accessible from [12] and [46].

As example, we report on the category *Vandalism*. Its second level includes *Damage of Vehicle*, which is further organized in *Break Window*, *Break Rearview Mirror*, *Throw Eggs*, *Throw Stones*, *Tire Puncturing*, *Scratch Vehicle*. It should be noted that the ontology can be further extended and tuned, for example to adapt to a specific application domain.

The classification of micro-events and macro-events following the ontology is also represented in the event model. Micro-events and macro-events are formalized by means of an appropriate and extensible set of metadata based on the information extracted from the analyzed resources. Figure 1 shows a simplified representation, for clarity purposes, of the UML class diagram of the event model. The whole event model can be downloaded from [46]. Classes represent the basic information of the model, while the arches represent the relationships between the different classes of the model. The event model is used to describe both the micro-events and macro-events. It also defines the relations between the different information that compose the micro-event or the macro-event. The *Event* is the central class of the model. The connected classes contain information on the event; they are linked to the event with space-time relations i.e., the place where it takes place, the date, the start time and the end time. The class *Category* refers to the corresponding classification following the earlier mentioned ontology.

We note the correlations (*EventRelationship*) and relations of causal (*CausalRelationship*), spatial (*SpatialRelationship*) and temporal (*TimeRelationship*) kind. The contents (*Resource*), regardless of their type, are extracted from different information sources (*Source*) and selected by means of pre-filtering processes which take place in a continuous manner. It should be noted that a *reputation* score can be associated to each *Source* (the reputation score can be used to rate the trustworthiness of a source [73]); consequently, events generated from sources with different reputation scores can be managed differently.

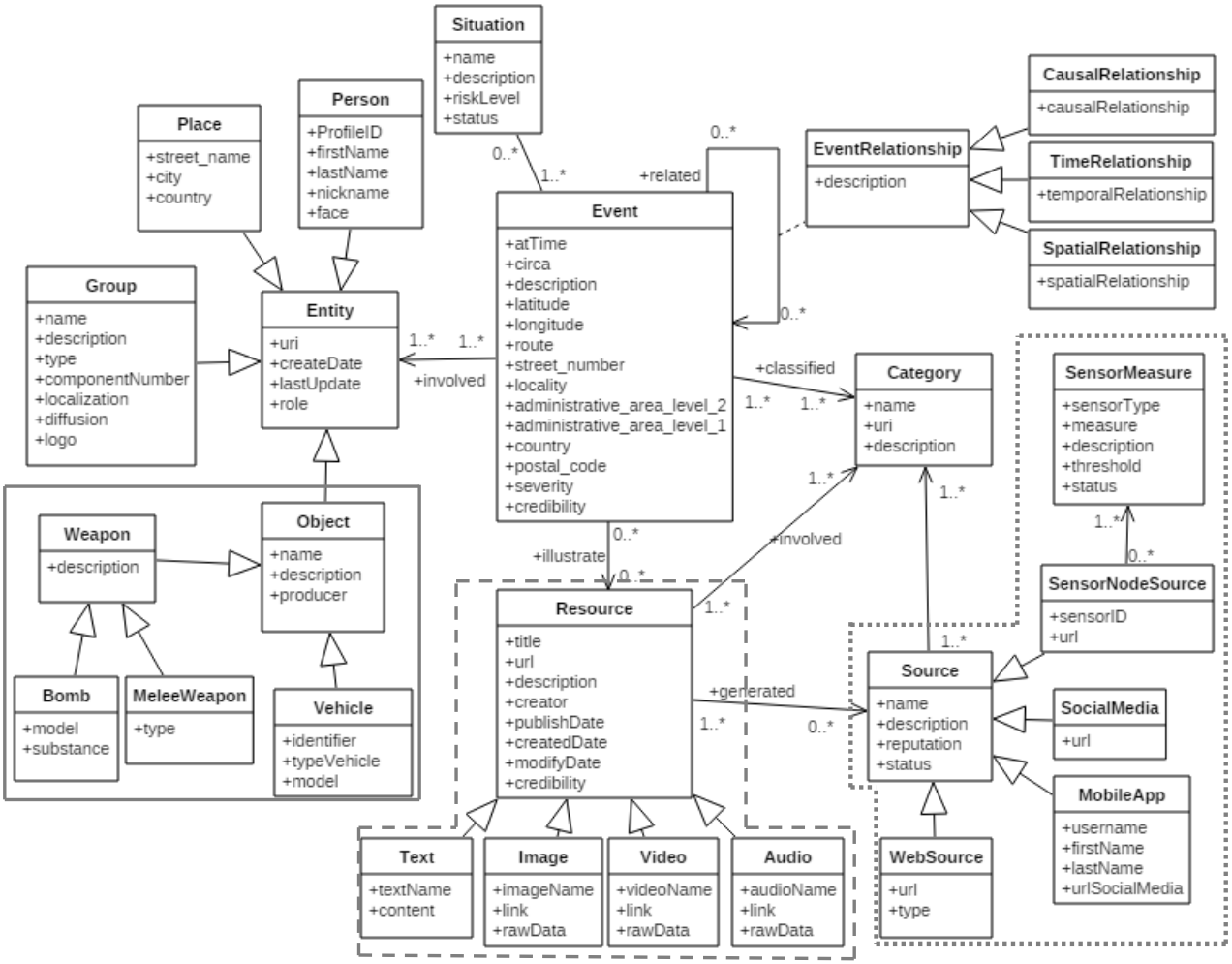
Different resource and source types are possible. Concerning resource types, in the dotted box of Fig. 1 we consider *Text*, *Image*, *Video*, *Audio*. Concerning sources, we consider the web, social media, sensors networks, and humans transmitting information through crowd-sensing mobile apps (e.g., see the application in [41]).

An event may involve (relation *involved*) a set of entities (*Entity*). We organize involved entities in: individuals (*Person*), place (*Place*), groups of people (*Group*), and objects (*Object*) of different types such as weapons (*Weapon*), or vehicles (*Vehicle*). The *Entity* is a fundamental component to describe an event and it is always present in each instance of the event model, including the case that the entity is not dangerous. For example, the *Person* entity may be *not* dangerous, but the one in danger: this distinction is clarified through the attribute *role* of *Entity*.

The possible entities are identified in the ontology; consequently, changing the ontology may lead to a different set of specializations of the *Entity*. For example, the Event Model of Fig. 1 assumes an ontology where only vehicles and weapons are the possible objects associated to an Entity (see the box on the left bottom corner of Fig. 1).

If a different ontology is used (e.g., an ontology intended to describe a specific crisis scenario), the *Category* and the specializations of *Entity* shall be updated.

Fig. 1. Overview of the Event Model.




When modelled using the event model, the micro-event is an event with only one *Entity* and only one *Resource*. Micro-events can describe a relevant entity e.g., people or objects that can be detected through processing of the information acquired from sensors. The macro-event can have more *Entities* and *Resources*. Macro-events describe critical situations: macro-events that describe the same situation are related in the event model through the *Situation* and the *Event* classes.

For explanatory purposes, Table I shows a short subset of the *micro-event* categories currently supported in the event model.

Table I. Sample micro-events.

| *micro-event category* | *description* |
|---|---|
| ASR- Abrupt Sound Recognition | Recognition of abrupt sounds related to critical situations (e.g., explosions, screams, breaking glass, intimidation) |
| PWOD- People With Object Detection | Detection of a person with a relevant object |
| SCB - Suspicious Crowd Behavior Detection | Identifies a set of suspicious behavior of people (or groups of people) in the scene that highlights abnormal situations, such as people that begin to run or congregate in a place |

Two examples of the *macro-event* category are instead shown in Table II, composed respectively of 2 and 3 types of micro-events. The vandalism example rule in Table II recognizes an act of vandalism when two *micro-event* types happen: i) *People With Object Detection* that detects a malicious person with an object that may be used to cause harm (e.g., explosive, knife, gun) and ii) *Abrupt Sound Recognition* detecting a sound due to critical events e.g., the breaking of a window, a car accident or a gunshot. The weapon attack rule recognizes a weapon attack through three specific *micro-event* types: i) *Abrupt Sound Recognition* to detect the explosion, ii) *Suspicious Crowd Behavior Detection* to detect people in panic, and iii) *People with Object Recognition* to detect the criminal. In practice, to describe a situation, the micro-events composing a macro-event must be collected within a specified temporal dis-

Table II. Macro- and micro-event types.

| *Macro-Event Category* | *Micro-event Category* |
|---|---|
| Vandalism | *PWOD & ASR*<br>one or more instances of PWOD and ASR detected within time interval $t_1$ and spatial distance $s_1$ |
| Weapon Attack | *SCBD & PWOD & ASR*<br>one or more instances of SCBD, PWOD and ASR within time interval $t_2$ and spatial distance $s_2$ |

tance $t$ and spatial distance $s$, and aggregated following specific rules (see Section 3.2).

### 3.2 Information Fusion Process

We describe the *information fusion process* that allows producing macro-events from micro-events in accordance to a specific event model.

The information fusion process is composed of three phases: *i)* syntactic check, *ii)* micro-event correlation and merging, and *iii)* verification of macro-event. It is described below and synthetized in Fig. 2. All the steps of Fig. 2 are executed online.

**Syntactic Check**

The first phase consists of an integrity check of the micro-events to verify their content, and search syntactic errors in their attributes. Examples are out of bounds in GPS coordinates, wrong format of date and time, presence of event categories not belonging to the particular application domain of the crisis management system. If the *micro-events* are syntactically correct, they are provided as input to the event correlation and merging phase. In the example of Fig, 2, all four micro-events pass this check.

**Event Correlation and Merging**

In the Event Correlation and Merging phase, rules are applied to correlate the micro-events received within a specific time frame. One or more micro-events are merged to compose one macro-event. In Fig. 2, micro-event 1 composes macro-event A, while micro-events 2 and 3 compose macro-event B. Micro-event 4 is not correlated.

A rule allows characterizing a specific *macro-event* type. A collection of rules detects all *macro-event* types in the particular application domain.

We still consider the example of Table II to discuss on rules. The macro-events of Table II result from the AND conjunction of categories extracted from the ontology, and of spatio-temporal distances. In order to compose macro-events, a rule is required that sets precise spatial and temporal conditions for the *micro-event* types; from now on, we will refer to these conditions as *delta* parameters. For both *vandalism* and *weapon attack*, the micro-events must occur in a specific temporal distance $t$ (e.g., less than 120 seconds) and spatial distance $s$ (e.g., less than 100 meters). The spatial conditions impose that the distance of the *Abrupt Sound Recognition micro-events* must be less than 1000 meters from the others. The temporal conditions impose that the delay from the *Abrupt Sound Recognition micro-events* to the others *micro-events* must be less than 120 seconds. In Table II, temporal order of the micro-events is not considered; in fact, our expectation is that micro-events are generated by different sources and consequently timeliness requirements on event generation cannot be set. New rules to compose macro-events can be defined, at the only condition that they are expressed using logical or arithmetical operators on categories and values.

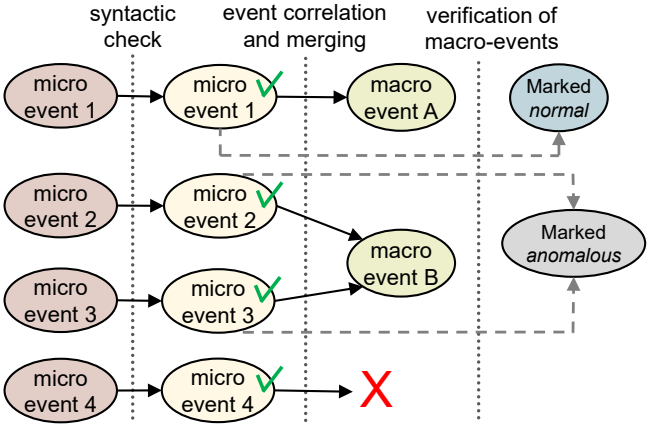



Fig. 2. The Information Fusion process.

**Verification of Macro-Events**

The identified macro-events could be affected by errors or uncertainties produced by human sources, or by correlation rules not sufficiently precise. In order to guarantee a consistent construction of the critical situation and a proper decision support to human operators, the macro-events must be checked to verify if they can be trusted. When a macro-event is produced, it is analyzed applying a trust analysis process (described in Section 4.2). This process observes the micro-events associated to the target macro-event; such micro-events are verified following different rules depending on the macro-event they are composing. At the end of this verification process, the macro-event is marked *normal* if it describes a critical situation, and *anomalous* otherwise.

In the example of Fig. 2, macro-event A is marked normal and macro-event B is marked anomalous.

## 4 Architecture of the Event Processor

The logical architecture of the event processor is shown in Fig. 3, where several components cooperate for managing, storing, correlating and aggregating events. The event processor is developed entirely in Java with the exception of the SQL database. Its components are Java RESTful web services running on Apache Tomcat 7.0 and their interfaces are described using JavaScript Object Notation (JSON). The event model is also implemented following the JSON format. The whole software developed is 11862 line of code excluding comments, white lines and Off-The-Shelf (OTS) libraries. Including OTS libraries, the whole size of the software is 302 MB.

We first describe the whole set of components, then we detail the Correlator (based on the CEP Esper [30]) in charge of performing the Event Correlation and Merging, and the Event Trust Analysis (ETA) component.

Micro-events come through the *Event Bus* that connects all the available *micro-event producers* to the event processor. The *micro-events producers* output data in a JSON notation which is compatible to the event model; consequently it can be used by the event processor.

Considering the bottom part of Fig. 3, examples of *micro-events producers* are from the left:

*i*) sensors network deployed to monitor an infrastructure. For example, in Section 6.3 a doppler radar to detect the

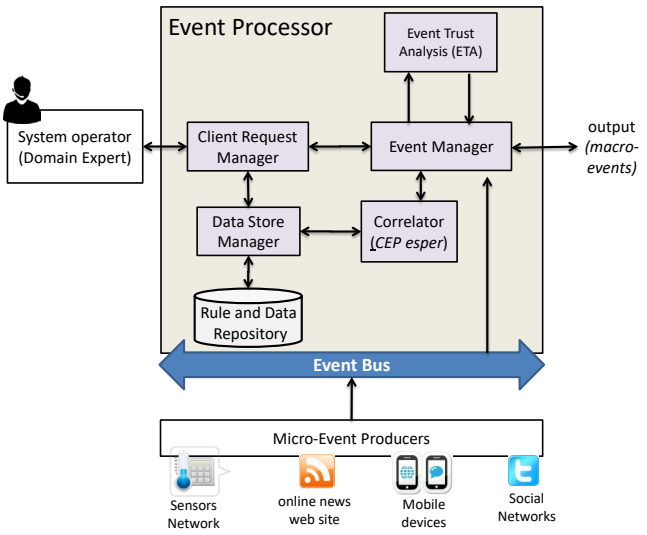



Fig. 3. Logical architecture of the event processor.

unexpected presence of aggregation of people and an accelerometer to detect vibration due to mass movements are set for surveillance of historical monuments;
*ii*) web site reporting online news. For example, in Section 6.3, news are extracted from websites to collect events regarding a football match and a political manifestation;
*iii*) mobile devices that send information via a specific application e.g., the Secure! App presented in Section 5 and applied in Section 6.3;
*iv*) social networks: messages extracted from Twitter are one of the information sources used in the case studies of Section 6.3.

The *Event Bus* is implemented by the Java Message Service (JMS). The Event Bus transmits all micro-events to the Event Manager.

The *Event Manager* acts as an orchestrator [23] which:
*i*) transmits micro-events to the Correlator for the event correlation and merging;
*ii*) transmits each macro-event and the micro-events that compose it to the ETA;
*iii*) outputs the macro-events. If the macro-event is marked *normal*, it contributes to depict the critical situation. If it is marked *anomalous*, it should be delivered to an operator which may accept, correct or discard it;
*iv*) coordinates configuration requests from the Client Request Manager to the Correlator and the ETA.

The *Correlator* correlates the micro-events to create a macro-event. It can be configured by operators which can define or modify correlation rules using the functionalities of the *Data Store Manager*. The Data Store Manager is the interface to access the repository where event model, ontology, correlation rules and, optionally, processed data are stored. The rules depend on the application domain in which the crisis management system is intended to operate; a different application domain can be selected just defining the specific set of rules.

The *ETA* checks anomalies in the macro-events produced: ETA verifies the consistency of the micro-events aggregated in a macro-event.

The *Client Request Manager* forwards requests from the system operator to the Event Manager or to the Data Store Manager. Sample requests are "add correlation rules", "update rules", "change configurations". The Client Request Manager offers to the operator a simple Java interface to define such requests. For example, we show in Fig. 4 the user interface to configure the Correlator. The left side shows the log file on the top, and the correlation rules on the bottom. The right side is devoted to the definition of new rules, which may be introduced uploading a file, or with the support of a menu and boxes to set the spatial and temporal parameters. Similar interfaces are available to configure the other services of the event processor. In general, configurations parameters are stored in the Data Store Manager. The interested services load these configurations when they are (re)started. The most significant configurations are: i) upload a new ontology or event model; ii) define new rules to compose macro-events; iii) modify the ETA parameters described in Section 4.2.

Finally, the Client Request Manager interfaces to the default administrator consoles [4], [40] that allow administering the REST services, the MySQL database, and the Apache Tomcat server.

### 4.1 The Correlator (CEP Esper)

The Correlator is a Complex Event Processor (CEP) implemented relying on the *Esper* [30] engine. Esper is an open source component based on *Event Stream Processing* (*ESP,* [30]) techniques. The Esper engine allows applications to store queries and run the data through [30]. The execution model is thus continuous rather than only when a query is submitted.

Rules are defined via Esper stream queries that provide the windows, aggregation, joining and analysis functions for use with streams of events. These queries are expressed through the Event Programming Language (EPL, [30]) syntax. Similar to tables in a SQL statement, EPL adopts the concept of *views*, which define the data available for querying and filtering. For example, views can represent windows over a stream of events. Also, it is relevant to notice that EPL makes possible to implement temporal conditions on incoming data. In our implementation, EPL rules are written in an XML file and stored in the Rule and Data Repository.

Esper was selected for our solution because it offers the following benefits: i) EPL rules can be modified at runtime without requiring to halt the event processor, and ii) Esper is totally developed in Java: for this reason it has been integrated easily in the event processor component like a Java library.

Esper uses listeners to apply the set of rules to the incoming micro-events. Each listener implements a specific set of rules and continuously applies the queries to all micro-events incoming from the Event bus. Every time a new micro-event reaches the event processor, the listener executes the queries.

In the event correlation and merging phase, the Correlator takes the received micro-events and correlates them based on space-time relations and according to the event

categories of the event model. Esper receives micro-events streams, applies EPL rules on them and returns the sets of micro-events that satisfy the rules.

We explain the performed correlation using the examples in Table III (for the meaning of the acronyms, see Table I and Table II), where two macro-events are matched to their correlation rules that are implemented in Esper.

The first rule aims to detect a vandal; the second rule aims to detect a weapon attack. The second rule could be wrongly activated also in different conditions. We can consider a case where the event processor wrongly interprets events as correlated, thus performing an incorrect detection (Weapon Attack). Let us consider a scenario of a running race in a city, and the simultaneous presence in the same area of an attacker accomplishing vandalism against a shop by breaking its window. In this case the event processor should recognize the act of vandalism without being influenced by the presence of people running. When the person breaks the showcase with a weapon, an *Abrupt Sound Recognition micro-event* and a *People with Object Recognition micro-event* are detected. In addition, a foot race passing through the interested area causes the detection of a *Suspicious Crowd Behaviour Detection micro-event*.

In this situation, the *vandalism macro-event* is correctly detected. A *Weapon Attack* is erroneously detected, because of people running close to the place where the sound is detected (distance is less than 1000 meters). The system misunderstands the vandalism for a weapon attack and outputs a false *Weapon Attack macro-event*. This wrong interpretation can be detected by the ETA, discussed in Section 4.2, which allows identifying potential anomalies in the events set.

Table III. EPL rule to create macro-events.

| Macro-Event | Micro-events | EPL rule |
|---|---|---|
| Vandalism | PWOD<br>ISR | select * from pattern [ every (event1=Event(event1.hasCategory(' PWOD') =true) and event2=Event (event2.hasCategory(ISR)=true)) where timer:within(320 sec)] where (event1.timeDiff(event2)<120 and event1.distanceGPS(event2)<100)) |
| Weapon Attack | SCBD<br>PWOR<br>ISR | select * from pattern [ every (event1=Event(event1.hasCategory('ISR')=true ) and event2=Event (event2.hasCategory('PWOR')=true) and event3=Event (event3.hasCategory('SCBD')=true)) where timer:within(10 min)] where(event1.timeDiff(event2)<120 and event1.distanceGPS(event2) <1000 and event1.timeDiff(event3) <120 and event1.distanceGPS (event2)<1000) |

Similar situations could also occur considering *cyber attacks* towards the crisis management system itself. An attacker could change data in the events transmitted to the event processor, causing the detection of a nonexistent critical situation, or avoiding the detection of real critical situations. In this context the spatial and temporal analysis of the ETA described in Section 4.2 could facilitate discovering cyber attacks in progress and warn the operator avoiding severe consequences.

Finally, we observe that a micro-event waits a time interval inside the Correlator for other micro-events to cre-

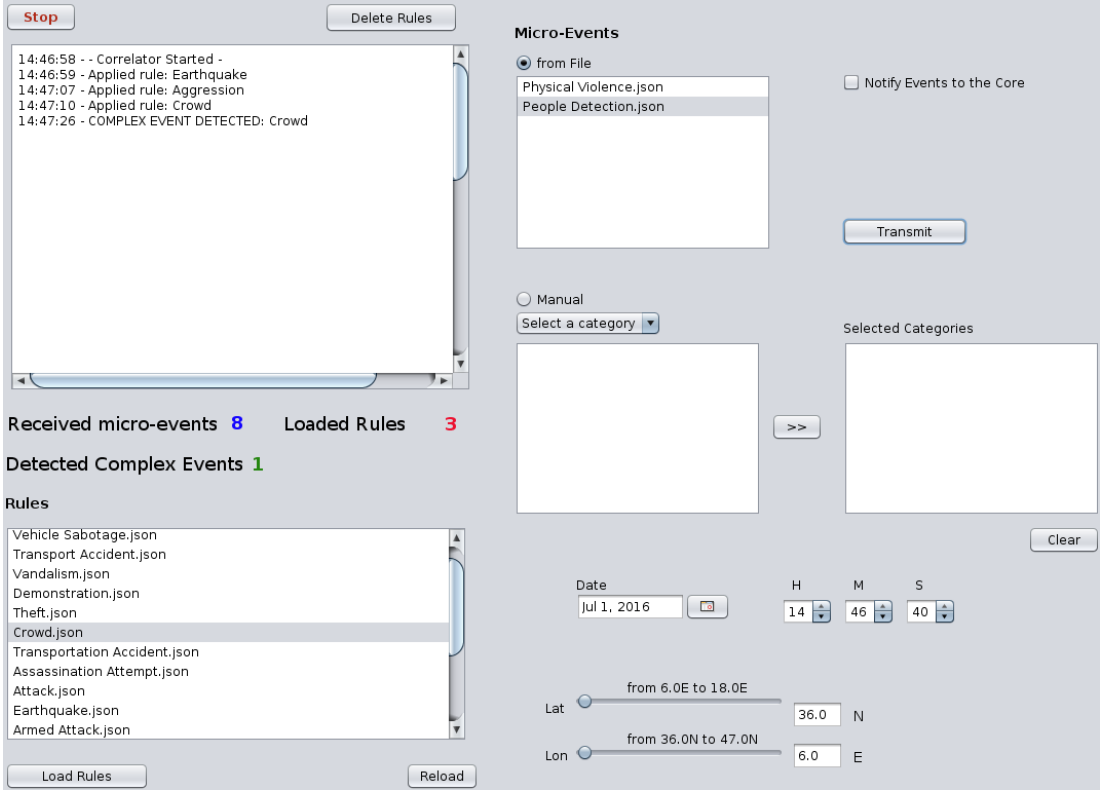



Fig. 4. User interface of the Correlator.

ate a macro-event. Considering as example *Vandalism* of Table II and Table III, the micro-event PWOD may wait up to $t_1$ (120 seconds in Table III) for a micro-event ASR; if ASR is generated within $t_1$, the macro-event *Vandalism* is created. All pairs of PWOD, ASR generated in a time window (320 seconds in Table III) contribute to the same macro-event.

### 4.2 The Event Trust Analysis (ETA)

The ETA component analyzes each macro-event created by the Correlator. ETA is able to detect the anomaly in the produced *macro-event* analyzing the spatial localization of the *micro-events* which compose it. The input of the ETA is the set of micro-events composing a macro-event; the output is the indication that the macro-event is *anomalous* or *normal*.

For example, considering the *Weapon Attack macro-event* previously discussed, it aggregates the three separated micro-events detected (*Abrupt Sound Recognition, People with Object Recognition* and *Suspicious Crowd Behaviour Detection)* including their own time and GPS locations. The *Weapon Attack macro-event* is recognized by ETA as anomalous because it recognizes two different micro-events clusters.

Before further progressing on the description of the internal algorithms of the ETA, it should be remarked that several possible algorithms exist and can be compared through benchmarks [24], [25], [31]. It is outside the scope of this paper to build a novel algorithm or compare existing ones to identify the most efficient. Different anomaly detection algorithms, not explored in this paper, may be selected for the ETA; main requirements that they have to meet are that they are able to cope with micro-events and they have adequate performance, operating at least at the same level of speed of the Correlator. Following these observations, we consider the work in [13] as a reference starting point, because it describes a process and algorithms that are able to meet the above demands. [13] operates on structured events stored in a repository, to identify relevant groups with similar attributes. We adapt [13] to deal with streamed data and to deal with the macro-event structured as in Figure 1. The process and algorithms we present here are able to provide an adequate (although most likely not optimal if compared with novel state-of-the-art algorithms) tradeoff between performance, false positive and false negative, as it is demonstrated in Section 6 through four use cases and further simulations.

#### *4.2.1 The Anomaly Detection Algorithms and Process*

**Assumptions**

For performing the anomaly detection we assume that more than $\alpha$% of the micro-events in the initial set are normal, i.e., the values of all their attributes are close to each other in time and location. Therefore we assume that for each attribute, their maximum distance from the mean value can be a predefined *delta* value. For example, events are normal if they happen within 10 minutes and their maximum distance is 1 km from their respective mean values. For the sake of simplicity, in our implemented algorithms we set $\alpha$=50%, thus considering that more than half of the set of micro-events must be normal in order to compose a normal macro-event. Without loss of generality in the methodological approach, the algorithms could be easily adapted to consider different $\alpha$ values. In practice, the parameters' setting should come out from the combination of i) the operator's experience in targeting specific crisis scenarios, ii) the expected operator-perceived trade-off between false positives and false negatives, and iii) the analysis of the available data stored in the Rule and Data repository to check the configurations used for detecting anomalies in similar situations.

```
algorithm eventFiltering

normalize the attribute values based on delta
for every attribute do
  divide the events into sets:
   width of each set is two*delta

  count the events in every two neighbouring set
    if numberOfEvents ≥ halfOfEvents
      then do nothing
    if numberOfEvents < halfOfEvents
      then mark events as anomalies
  end for
```

Listing 1. Pseudocode of the event filtering algorithm

**Event filtering algorithm**

The event filtering algorithm (Listing 1) detects sporadic events with large magnitude differences in at least one of their attributes. Based on the *delta* value, we search for those intervals in which the majority of the events are located (for each attribute). Events that are not in these intervals are considered anomalies. This is a rough, but very fast algorithm. By selecting large intervals (a large *delta*) we can be sure that normal events will not be detected as anomalies.

**Mean value algorithm**

We can detect normal events by computing the mean value for all the attributes separately, and checking if the distance from this mean value is below the predefined *delta*. If more than half of the events are normal, then the remaining events are detected as anomalies, and the anomaly detection is complete. This is the mean value algorithm (see Fig. 5, Listing 2).

**K-medoids algorithm**

If the anomalies form separate clusters, the mean value of the set is located far from the normal events and the mean value algorithm is not successful.

We use the k-medoids algorithm to detect the clusters, and then check the largest cluster with the mean value algorithm. The algorithm partitions the events into *k* clusters by selecting *k* events as medoids and assigning all the other events to the closest medoid. For a detailed description, see [14].

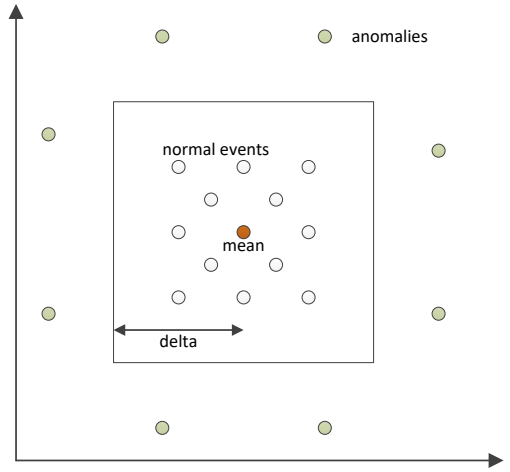



Fig. 5. Graphical representation of the mean value algorithm.

**The resulting process**

The logical information workflow of ETA is depicted in Fig. 6. It contains three main modules: the event filtering algorithm, the mean value algorithm and the k-medoids algorithm. These three modules are of incremental computational cost.

The algorithms run in a specific order. First, the event filtering algorithm is executed with a large magnitude difference. Then the mean value algorithm executes to see if more than half of the events are normal. If this is the case, the algorithm stops, and lists the other events as anomalies. Otherwise, clusters are searched using the k-medoids algorithm until a limit value $k_{max}$. After the identification of clusters, the largest cluster containing more than half of the events is checked using the mean value algorithm, stopping the analysis when the normal (and then anomalous) events are found. The clustering can be repeated for different $k$ as the number of clusters, until $k$ reaches $k_{max}$.

This order guarantees that only the necessary tasks are performed to reach the goal. The event filtering is very fast, especially if the events are close to each other. By running event filtering in the beginning, we can improve the success rate of the simple mean value algorithm, which can be negatively affected by the presence of sporadic events. The mean value algorithm should produce a result in most of the cases, leaving the k-medoids algorithm for special and rare cases (high computational cost).

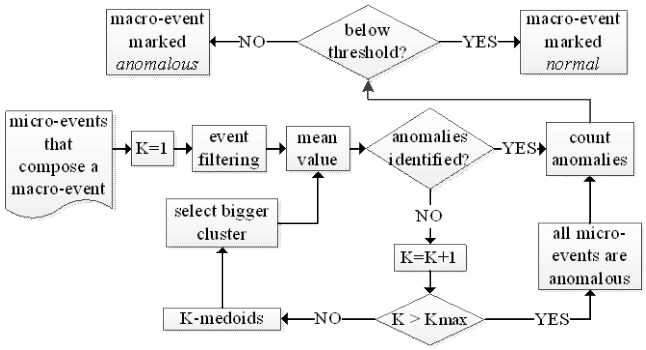



Fig. 6. Information workflow for ETA.

**algorithm meanValue**

```
for every attribute do
  compute mean value of the attribute
  for every event
    if (current attribute value > (mean value + delta)
    or current attribute value < (mean value – delta)
      then mark event as anomaly
  end for
end for
```

Listing 2. Pseudocode of the mean value algorithm

If the percentage of micro-events detected as anomalous is above a determined threshold, the macro-event is labeled as anomalous, otherwise as normal. The maximum value for the threshold is 50% i.e., if more than half of the events are anomalous, the macro-event is always marked anomalous.

At the end of its execution, the ETA rates the macro-event as anomalous if the percentage of anomalous micro-events is above a determined threshold, and normal otherwise. The anomalous micro-events do not contribute to compose the normal macro-event.

From the perspective of the operator, the ETA workflow is considered as a black box. The intermediary steps are hidden to the operator, which has at his disposal only the final outputs. This is a necessary design choice, because the continuous execution of the event processor and the ETA is independent from the responsiveness of the operator.

## 5 Application in a Crisis Management System

The Secure! framework was developed in the context of the recently concluded Secure! project [6]. It is a Decision Support System (DSS) for crisis and emergency management. It exploits information retrieved from different types of sensors deployed in the area of interest in order to detect critical situations and determine the corresponding reaction. Users have the opportunity to directly interact with the Secure! framework using their mobile devices: they can provide information about real events and receive information about critical situation in which they could be involved. The Secure! framework is also intended to detect critical situations before they happen: it analyzes real events provided by the social media and correlates them with historical data and events incoming from other sources. For example, threats to people or things may be detected making a syntactic analysis of the text content provided by social media; searching for particular keywords, it is possible to recognize the intentions of a spiteful person.

Input data may come from several sources, for example: i) social media, ii) web sites, especially those which report news, iii) mobile devices, via a Secure! App that is available to users, iv) sensor networks deployed in buildings or critical infrastructures.

The logical architecture of the Secure! framework is organized in four distinct levels, each of which comprises

logical components and services. Starting from the bottom layer, they are:
*(i) Source Data Collector and Media Integration*. It collects heterogeneous micro-events from multimodal sources connected to the system, and it extracts information from the collected data. The possible approaches to extract the data from sources and convert them in micro-events are not debated in this paper. More information on the technologies applied in Secure! can be found in [45].
*(ii) Event Extraction and Integration*. It generates events that describe the critical situations.
*(iii) Situation Extraction and Awareness*. It is a human-assisted process where macro-events and historical data are presented to support coordinating the crisis management.
*(iv) Secure! Apps and Services.* These represent the interface between the Secure! framework and the Secure! users. Sample users are operators in operative centers, and operators on field equipped with ad-hoc devices. Also, registered users (including civilians) can install and use the *Secure! App* on their mobile devices to notify events to the Secure! framework.

Our work has been instantiated in the *Event Extraction and Integration* level of the Secure! logical architecture where correlated events are used to build the critical situation.

The event model (Fig. 1) is applied in Secure! to represent micro-events and macro-events. Once micro-events are received from the Source Data Collector and Media Integration layer, the event processor processes and aggregates the input data to create macro-events. Afterwards macro-events are checked by ETA; anomalous ones are identified.

In the layer Situation Extraction and Awareness, the *normal* macro-events are directly used to support the analysis. The anomalous events are maintained separate, until the human operator decides to manually accept, correct or eliminate them. This approach exploits the synergy between the power of the automatic anomaly detection and the human cognitive ability to recognize errors or inconsistencies in the events.

## 6 Tests and Validation

This section presents the validation of our event processor, using reference scenarios and dataset defined during the validation process of the Secure! framework.

The event processor is installed on 2 virtual machines (VMs), one entirely devoted to the ETA, and one to the remaining components of the event processor. The reason of this setting was to separate the ETA and the Correlator in order to easily measure their individual performance. The two VMs are configured with 2 GB RAM running under a physical machine with an Intel Xeon E5-2620 (6 Cores) CPU 2.00GHz processor with 48 GB of RAM. The remaining services of the Secure! framework are connected via RESTful web services and instantiated on separate machines not described here.

The event processor has been tested and validated experimenting it with a set of correlation rules and a set of input events conforming to the Secure! domain.

A large set of functional tests have been performed on the event processor. These tests were performed with specific inputs that allowed to stimulate the different functions of the event processor and to exploit the application of the different rules. The objective of these tests was to verify the functional behavior of the event processor and detect possible software bugs.

The rest of our activities consisted in the following tests to validate the event processor and the ETA in simulative and real settings. Performance tests in Section 6.1 show the computational time of the event processor. They use a synthetic input dataset comprising micro-events created with a micro-events generator. The ETA was also tested to verify the accuracy of the detection: these tests are reported in Section 6.2. Section 6.2 relies on a synthetic input dataset comprising micro-events from Table I created using a micro-event generator. These micro-events allow the generation of the macro-events in Table II. Finally, the whole event processor was exercised within the Secure! framework in realistic scenarios. This included i) feeding the event processor with data collected during past emergencies, and also ii) feeding it with data collected on the field at runtime, by dispatched sensors. These tests exercise the event processor on real data collected from the field. Micro-events are generated from web sites, sensors, tweets, *Secure! app*, and local civil protection. The outcome of this test is reported in Section 6.3.

The dataset is publicly available for download at [46], together with the adopted ontology, the event model and correlation rules, and the simulation tool developed for Section 6.2.

### 6.1 Performance tests

The purpose of these tests is to measure the performance and the scalability of the event processor (without ETA) and of the ETA. A large number of input events (micro-events) are generated for every test case using a micro-event generator module. This module generates micro-events randomizing, within a preset range, the attributes *latitude*, *longitude*. They obviously have other attributes, for example the category and the generation time. For the performance testing considered in this section, sets of 100, 500, 1000, 3000, 7000 and 10.000 events were generated, once without clusters, and once with two clusters. The events set with two clusters contain one cluster with approximately 55% of the events and one cluster with approximately 45% of the events.

After the generation of micro-events, the second step is to run the correlator and create the macro-events based on spatio-temporal distance and their category.

The third and final step is to run the ETA to detect the anomalies. The performance of this component depends on the necessary algorithms, meaning that for test cases containing clusters we presume that the component will need more time to complete the analysis. The distance between the two clusters is not measured by the event processor, because the correlation rule is satisfied by pair of events.

We measured the processing time of the event proces-

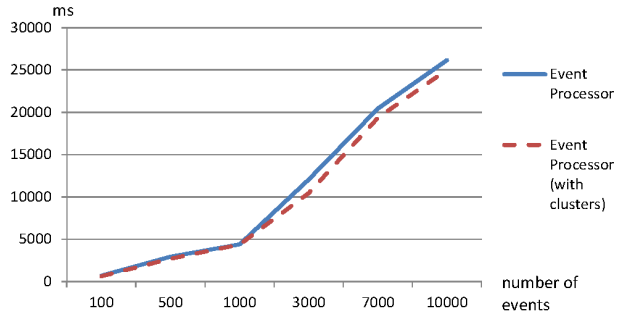



Fig. 7. Performance of the event processor without ETA (axes in logaritmic scale).

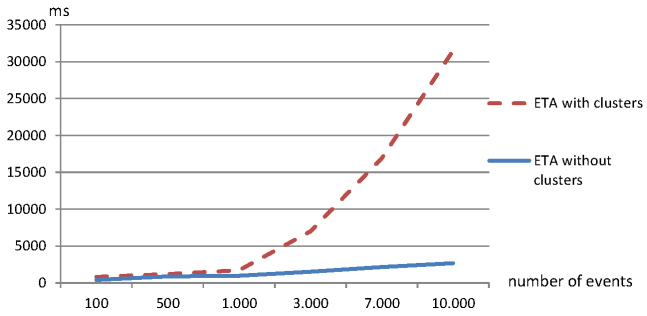



Fig. 8. Performance of the ETA component (axes in logaritmic scale).

sor (without ETA) and the ETA. For ETA the timer started after loading all the macro-events from the files. For every test case the processing time is calculated as the average of three separate runs.

The processing time of the event processor resulted independent of the clusters, and close to linear for the number of events (see Fig. 7).

The processing time of the ETA highly depends on the presence of clusters, especially for large number of events (see Fig. 8). The time difference is completely justified by the running of the k-medoids algorithm (together with the building of the distance matrix). Another result is that the anomaly detection (if the mean value algorithm is successful) is significantly faster than the rest of the event processor, and especially the Correlator, for large number of events. In general, the ETA is significantly quicker than the Correlator, which is a fundamental requirement to successfully use the ETA. The performances of the ETA fall behind those of the Correlator only when the clustering is applied to a high number of micro-events (approximately above 7000). We believe this is a limit condition which is expected to happen rarely.

The *individual* micro-events [46] provided as input are representative of potential micro-events generated in a real deployment. The *stream* of micro-events, instead, is intended to investigate the computational performance of the event processor: consequently, depending on the considered scenario the number and the frequency of arrival of micro-events may be unrealistic. For example, in Section 6.3 the event processor is exercised to observe local areas, and the number of input micro-events is in the order of hundreds per hour. Compared to Fig. 7, we can observe that the event processor with its current computational resources is fully able to manage the inputs of Section 6.3. In other scenarios, for example when a larger area is observed, the event processor may have to process a higher number of micro-events, leading to a higher computational load.

## 6.2 ETA Validation

### *6.2.1 Validation methodology using R and RStudio*

By creating interface points between the anomaly detection process and our validation tool, the processing of the events can be tracked as the ETA algorithms run after each other. In fact, we defined a tool that loads the data (macro-events), runs the algorithms on the ETA, and shows the results in diagrams. The data produced during this process can be exported from the ETA at any point (meaning before or after every algorithm). This way the different algorithms can be checked individually.

To create such tool, we relied on the open source statistical environment R [15]. We used RStudio open source IDE developed for R [16] to write and run the R programs used for validation. We used the *pam* function of the cluster [17] package which is a robust implementation of the K-means algorithm, and auxiliary packages [18], [19], [20], [21] to visualize the data on 2D and 3D plots. The tool developed is available at [46].

### *6.2.2 Validation of the ETA detection capability*

For every test case, the assumption on the input events has to be valid, meaning that more than half of the events are *normal* events. The goal is to measure the number of false positives/negatives in case of different setting of the ETA.

*False positive* event is a normal event detected as anomalous. *False negative* event is an anomalous event detected as normal.

Fig. 9 shows the test case used for the analysis. The events located in the center area are normal, while the events in the corner areas are anomalies. Within each area the events are randomly distributed. The areas and number of events were set in order to use only the mean value algorithm for the detection; in this case the runtime execution only depends on the number of anomalies and not on the usage of the clustering algorithm. This test case can actually resemble a scenario of a running race in a city (generating 800 events within the central area) and the simultaneous presence of an attacker accomplishing a robbery against a shop breaking its window (in any of the neighboring areas, generating 50 events).

As already discussed, the set of micro-events actually belonging to two different situations (Vandalism and Foot Race) can be wrongly interpreted by the event processor as correlated. This results in incorrect macro-event detection (Weapon Attack). This misbehavior can be properly detected by the ETA, and its accuracy in terms of false positives and false negatives depends on the assigned *delta* value.

We explore the validation of the ETA varying the *delta* parameter. In fact, this is the most relevant parameter for detection purposes, which is at the basis of the mean value and the event filtering algorithms.

The left side of Fig. 10 shows the number of false positive events for several values of *delta* (from 78% to 100%).

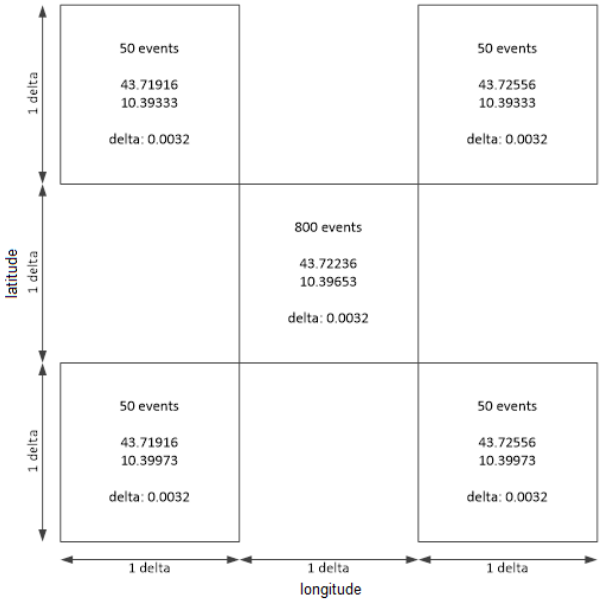



Fig. 9. Test case used for sensitivity analysis.

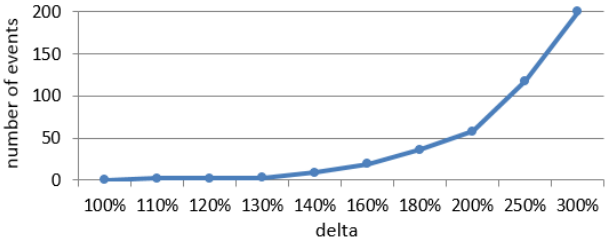



Fig. 11. Number of false negative events with respect to the increased values of *delta*.

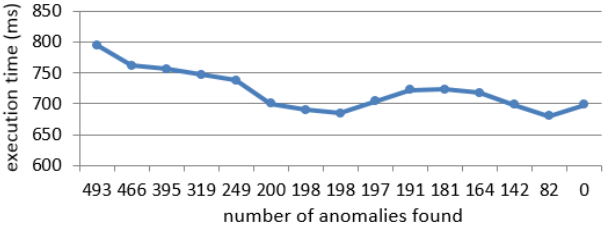



Fig. 12. Runtime of ETA with respect to the number of anomalies found.

The event set contains 800 normal events (e.g., related to the foot-race) and 200 anomalies (e.g., those related to vandalism attacks). At least 501 events have to be detected as normal; meaning the maximum number of false positive events can be 299. If we set 78% of the original *delta* for the analysis, the number of false positive events is 293. Nevertheless, all true anomalies (e.g., those related to the vandalism attacks) are correctly detected. The right side of Fig. 10, and Fig. 11, show the number of false negative events for several *delta* setting (from 100% to 300%). Since the location of the original events is not modified, multiple runs of the algorithm have the same result.

During the sensitivity analysis the performance of the ETA component was also measured. For every test case the average of three runs are shown in Fig. 12. The result shows that the number of anomalies does not significantly affect the runtime of the component, as expected.

The sensitivity analysis of the ETA component shows the importance of the *delta* parameter used for anomaly detection. This result emphasizes the fact that an anomaly is defined "as something which is not normal". If we soften the conditions on "what is normal", then no anomalies will be found.

Last, we comment on the $K_{max}$ and *threshold* parameters (see Fig. 6). Different values can be assigned to the $K_{max}$ parameter. In our case study, for $K_{max} \geq 5$ we only observed minimal differences that consequently we do not report here. Setting $K_{max} < 5$ obviously fails to identify all the clusters in Fig. 9. Finally, we comment on the parameter *threshold*, considering our set of 1000 micro-events and the anomalies identified in Fig. 12. If the threshold is lower than *1000 - number of anomalies*, the entire macro-event is identified as normal, otherwise it is labelled anomalous. For example, in our case study, at most 493 anomalies are identified (see Fig. 12): if the threshold is lower than 1000-493=507, the macro-event is always marked normal.

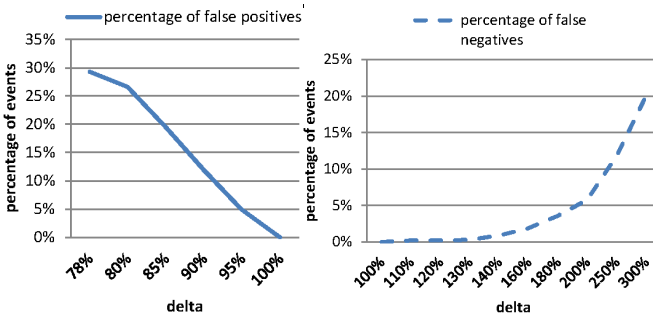



Fig. 10. Percentage of false positive (left) and false negative (right) events with respect to the decreased and increased values of *delta*, respectively.

### 6.3 Realistic scenarios and a real deployment

The event processor was tested in realistic settings after the integration with the whole Secure! system. Leaving aside integration tests and functional tests, we illustrate its application in *three selected scenarios* and a *long run* of the whole Secure! system to monitor the famous *Piazza dei Miracoli* of Pisa (the scope of the crisis management system in this case is the world heritage protection).

These experiments bring evidence that the event processor is able to operate with real data extracted from the field. Additionally, they provide clear indications on the dimension of the input streams that should be expected for local ad-hoc deployments.

The three scenarios were tested a-posteriori. In fact, information produced during the emergencies were collected a posteriori, and submitted to the Secure! framework. The three scenarios are:

- *Europa League Match* (night between 18th and 19th February 2015): clashes and vandalism between supporters and police in Rome occurred before the Roma - Feyenoord European football match. 845 micro-events were available from: 1 note from the police headquarters warning on potential risks connected to the match, tweets trends, web site of online news. This scenario also included 5 messages from Secure! App (human sensors) notifying clashes occurring in

Piazza Campo dei Fiori, where the largest part of the disorders occurred. These three messages contributed to generate three of the 22 macro-events that refer to that area of the city. In total, 64 normal macro-events were generated that described the evolution of the situation through time.

- *Political Manifestation*: clashes between police and violent members of Italian political factions (*Lega Nord*, *Casapound*) at a manifestation in Rome on the 27th and 28th February 2015. In this case, 30 normal macro-events were collected out of 630 input micro-events composed of tweets, and web site of online news.
- *Weather Warnings*: intense atmospheric events (strong winds) raged in Tuscany on the 4th and 5th of March 2015. Micro-events inputs where tweets trends describing weather conditions in locations of Tuscany, events from the local civil protection agency (it periodically dispatches updates on weather status in case of bad weather) and web site of online news. In total, 44 normal macro-events were generated in the abovementioned time window.

Finally, in a long run of approximately two months, the Secure! system and the event processor were exercised to monitor the UNESCO site *Piazza dei Miracoli* (Square of Miracles) in Pisa (Italy). The Secure! system was configured to provide to the event processor micro-events originated from i) the twitter trends collected from Twitter, ii) data on movement of people collected by a doppler radar, iii) vibration data from an accelerometer. The doppler radar and the accelerometer were transmitting fresh data to Secure! at the rate of 1 update per second.

Despite relevant critical situation did not happen in Piazza dei Miracoli for the duration of the considered experiment, the event processor created 612 macro-events distributed over the two months. These macro-events were classified following the ontology as "public gathering", and they are due to the activity of the doppler radar and accelerometer. In fact, these are intended to detect aggregations of people (masses of tourists) moving altogether: the doppler radar and accelerometer measure changes in the movement of groups of people. Each of the two sensors raises an alert when the measured value is above a defined threshold. This leads to the generation of a micro-event of type *people detection* that is provided to the event processor. If *people detection* micro-events are generated from both sensors in a time window, the corresponding macro-event is generated.

It should be noted that according to [56] the yearly number of tourists entering Piazza dei Miracoli is approximately 2.5 million, implying that large aggregations or unexpected movements are expected. A different threshold of the sensors could reduce the number of false alarms. This is a potential enhancement for the usage of Secure!.

We conclude the discussion mentioning that novel sources could be introduced at the only cost of modifying the event model to appropriately include them. Also, novel ontology and rules could be introduced. In the specific case of Secure!, micro-events are generated from the layer *Source Data Collector and Media Integration* which lays below the event processor: the modified event model and the ontology must be shared between such layer and the event processor itself. Instead new rules can be added through the configuration interface of the event processor.

## 7 Related Works

Several studies describe enabling approaches, mechanisms and supporting tools for crisis management, spanning on different contexts and involving social media for data acquisition and alert diffusion [38]. We organize the literature review as follows. In Section 7.1 we discuss the ontology and the event model: we consider alternative ontologies, and the possible compliance of our event processor to existing event model. In Section 7.2 we compare our event processor with existing solutions for crisis detection and management. Finally, in Section 7.3, we review the usage of anomaly detection in crisis management.

### 7.1 Ontologies and event models

Several ontologies specifically devoted to crisis-oriented and disasters characterization are available [38], [47]. Focusing on those intended to describe disasters and critical events [47], most relevant ones are ISyCry [37], SOKNOS [50], WB-OS [51], EM-DAT [54], Canadian Disaster Database [52], Australian AG Disasters Database [53], Humanitarian eXchange Language HXL [55]. Unfortunately, these are either i) formally represented but not available publicly ([37], [50], [51]), or ii) they are intended for online querying and not available for download in a standard representation language for ontologies ([52], [53], [54], [55]). This led us to define a novel ontology, which is i) freely available for download [12] and ii) represented in the well-known language OWL. It should be further remarked that novel ontologies, or extension of the existing one, can be introduced in our event processor as already discussed in Section 3.1: our ontology shares the key concepts common to most of the abovementioned ontologies.

Events are often linked to other concepts or meta-data that are fundamental for the definition of the crisis. It is relevant to discuss solutions from the state of the art to build events starting from sources.

This allows verifying that our event model (and event processor) i) can be used in a wide range of contexts, and ii) can be integrated to existing data extraction tools. In fact, many authors faced the issue of crisis characterization using a model-based approach. Noteworthy, our event model is devoted to describe the critical event, and consequently event models for management as [39] are not considered.

We describe our findings with the support of Table IV. The first column enlists the surveyed works. The second column *cat.* classifies the works in four categories (note that some works may comprise different categories) [38]:

TABLE IV. EVENT MODELS FOR CRISIS MANAGEMENT AND POSITIONING OF OUR WORK

| | cat. | use case | domain | sources | event representation | real-word deployment | Compatibility with our approach |
|---|---|---|---|---|---|---|---|
| *our work* | *CMS* | *see Section 6.3* | *crisis domain* | *Twitter, news sites, sensors networks* | *UML schema supported by an OWL ontology* | *Piazza dei Miracoli, Pisa* | |
| [66] | IC, IS | several: flooding, bombing, … | crisis domain | microblogging (Twitter) | a crisis lexicon | no (a posteriori) | the lexicon can be the input ontology for our event processor |
| [67] | IC, IES | Joplin 2011 Tornado | crisis domain | microblogging (Twitter) | machine learning for posts classification, based on an ontology | no (a posteriori) | the tailored ontology can become compatible with our event model |
| [68] | IS, IC | Hurricane Sandy, 2012 | crisis-specific (hurricane) | news on Reddit.com | content type and formats of most relevant Sandy threads | no (a posteriori) | Reddit is a possible input source |
| [69] | IC | Haiti 2010 earthquake | crisis-specific (earthquake) | text messages (Ushaidi dataset) | a vocabulary of words and labels to classify and aggregate messages using machine learning | no (a posteriori) | can create micro-events (from text messages sources) |
| [70] | IS | crimes in Monterrey, Mexico, in 2010 | crisis-specific (crime) | microblogging (Twitter) | Tweets content and related metadata including relevance of the sources | yes | micro-events can be generated from the metadata considered. W.r.t. ETA, it provides trust information in a semi-automatic fashion. |
| [71] | IES | not reported | crisis domain | text from Web documents | includes structured description of events | no | generates updates of ongoing crisis that could be micro-events (textual sources) |
| [72] | IES | several from different domains | general purpose | microblogging (Twitter) | create clusters but a specific event model was not identified | no (a posteriori) | tweet summarization can generate micro-events, or event complex *tweets-only* macro-events |
| [73] | IC, IS | TREC microblog benchmakr | crisis domain | microblogging (Twitter) | tweet with semantic enrichment: involved entities, classification of the tweet, links, user metadata | yes | enriched tweets can be used as micro-events |
| [48] | CMS | forest fires | crisis-specific | not specified, provided through a User Interface | follows an ontology specific to the definition and monitoring of plans for forest fighting | not specified | the ontology can be the basis to build our micro-events |
| [58] | CMS | Earthquake in Pakistan, 2013 | crisis domain | microblogging (Twitter) | Relies on labelling categories: crisis-specific labels from previous similar disasters | yes | the labelling categories can be introduced in our event model |
| [62] | CMS | Haiti 2010 earthquake, BP oil spill, 2010 | crisis domain | Twitter + potentially other sources | geographic, temporal, and thematic information | no (a posteriori) | the required data and metadata are compatible to our event model |
| [61] | CMS | 2012 Syria civil war | crisis domain (large scale events) | microblogging (Twitter) | based on keywords to organize Tweets; includes metadata on the tweet | yes | |
| [42] | CMS | not reported | crisis domain | sensors networks | ontology available for dependability, capability, system assessment | no (demo only) | the ontology is not adequate for our solution because it does not describe the event but only the system status |

- *information content* category *IC*: works that extract categories of information from the input data;

- *information source* category *IS*: works that select messages or data from specific groups;

- *information extraction and summarization* category *IES*: works that focus on subdocument analysis and information extraction and summarization.

- *crisis management systems CMS*: works that include the whole event processing activities to describe a crisis, and possibly also address the management phases. These works are competitors of our solution (differences are in Section 7.2).

The successive columns of Table IV report i) the *use cases* adopted for the considered work, and ii) the *reference*

*domain* distinguishing between general purpose, crisis domain, and crisis-specific. *General purpose* means that the domain is not restricted to crisis management; the *crisis domain* includes different contexts but always related to crisis; the *crisis-specific* refers to a specific crisis context.

Table IV also reports the *sources*, the *event representation*, and information on *real-world deployments* (*yes* if the solution in the considered work is exercised during a real crisis, *no* if the validation was performed a-posteriori using data from past crises). Finally, a judgment on the *compatibility* of the considered event representation and solution with our ontology, event model and event processor is reported.

From Table IV, it results that our work is an original one because of its capability to integrate in existing data extraction tools. Also, several event models are intended for only one source or only a well-defined set of sources. Additionally, several event models are crisis-specific. It should be noted that our work can be used in conjunction with most of the reviewed ones, at the cost of modifying the ontology and event model as discussed in Table IV column *compatibility with our approach*.

### 7.2 Crisis Management Solutions

The integration of information from different sources is an effective mean to improve situational awareness in crisis management systems [64]. Our work focuses on strategies and technologies for the analysis of information collected by different sources, including social networks and sensors network, as well as investigating the accuracy of the information through the identification of geo-spatial anomalies.

We compare our work to the most relevant near real-time CMSs we identified in literature. We identify main differences and novelties of our approach. Noteworthy, some of the works considered in the following are not reported in Table IV, because they did not place adequate emphasis on the event model or target ontology.

We first review *crisis-specific CMSs. In general, crisis-specific CMSs cannot be easily applied to other contexts, thus differentiating from our solution*. Examples are [48], [59]. SIADEX [48] integrates AI techniques to design fighting plans against forest fires. It analyzes the knowledge stored in an ontology server and it creates an attack plan, whose execution is successively monitored. [59] is a framework for real-time humanitarian logistics data focused on mathematical modeling. It is deployed and managed manually to meet specific requirements of a post-crisis management.

*Several CMSs address the broad crisis domain but they focus on only one source of information. The objective of our event processor instead is being able to manage multiple sources, merging data from social media, sensors networks, alerts dispatched from authorities, online news, etc*. AIDR (Artificial Intelligence for Disaster Response [58], [74]) is a platform designed to perform automatic classification of crisis-related microblog communications. It performs a classification of tweets generated during large-scale disasters. The Emergency Situation Awareness (ESA, [65]) collects tweets from Australia and New Zealand and processes them to identify unexpected incidents, monitor ongoing emergency events and provide access to an archive to explore past events. CrisisTracker [61] is an online system that efficiently captures distributed situation awareness reports based on Twitter activity during large-scale events, such as natural disasters or wars.

*Other CMSs, both crisis-specific and domain-specific, consider different sources but do not tackle the issue of heterogeneity. Including new sources is not a central issue of their design*. For example, the following CMSs use a known number of sources, mainly deployed sensors networks or ad-hoc personal devices. [33] describes an architecture for planning and decision support for environmental information management. It relies on dispatched sensors able to provide vehicle position and weather information. [34] delivers tsunami warning messages using information about tide gauge, seismology, GPS ocean observation, weather collected from dispatched sensors. CodeBlue [27] is a distributed architecture comprising wireless sensors and remote devices that allows monitoring and tracking of patients and first responders into disaster response scenarios. The remote devices are wearable vital sign sensors, handheld computers, and location-tracking tags. The RT-HRLE [49] system uses a wireless sensor network for real-time monitoring, tracking of missed people inside buildings and reporting partial or total destruction of buildings. [42] presents an approach for architecting wireless sensors networks and connected crisis management system. The resulting sensors network can provide resilient search and rescue capabilities.

Finally, we devote special attention to SensePlace2 [62]. Although it is specialized to acquire and process data from Twitter, it is conceived to consider alternative *text-based* sources. SensePlace2 is a web-based geovisual analytics application through which information contained in social media can be gathered and analyzed to support situational awareness in crisis management and related application domains. However, *w.r.t. our work, SensePlace2 is focused on text-based sources and it is leaning towards geo-localization and visualization rather than on merging data from different sources to detect and depict a situation.*

We conclude this discussion reporting on the usage of CEP technologies to manage large data streams. CEP-based technologies have been widely used for managing streams of data, and especially in the context of crisis management systems [26]. Their adequacy for managing large stream of data has been documented before. For example, in [29] a distributed CEP system integrates different CEP engines, and it is applied to a nuclear crisis management scenario. In [32], a Mediation Information System (MIS) is proposed to help the crisis cell partners to design, run and manage the workflows of the response to a crisis situation, where a CEP engine is used to consume and manage events. *To the best of our knowledge, CEP has never been used to create macro-events from micro-events as proposed in our solution. However, the state of the art on CEP technology here reported makes us confident that CEP is adequate to operate on the data stream managed by the event processor.*

### 7.3 Anomaly Detection and Crisis Management

Several technologies exploiting crowd-sensing and crowd-sourcing information fusion for situational awareness are described in the literature. However, being able to verify the quality and trust of the information, which in several cases is left to the users (as in the Ushahidi platform [9]), is currently an open challenge. Several approaches verify the trustworthiness of the information sources [10], but they are not considering heterogeneous data sources for detecting a critical situation. In this paper we consider heterogeneous sources of information and we exploit anomaly detection techniques to identify anomalous (i.e., not trustable) events actually not belonging to the identified critical situation.

Anomaly detection refers to the problem of finding patterns in data that do not conform to the expected behavior [31]. Anomaly detection techniques are extensively used in a wide variety of applications such as fraud detection for credit cards, insurance or health care, intrusion detection for cyber-security, fault detection in safety critical systems, and military surveillance of enemy activities [31].

Regarding crisis management systems and anomaly detection, [10] proposes an anomaly detection technique against the diffusion of false information through social media and activities linked to compromised accounts. The proposed solution is based on an analysis process that models the behavior of normal users accurately and identifies significant deviations from it as anomalous. [11] analyses *tweets* using a cluster analysis approach to distinguish between local event reports and global media reaction, and to detect spatial-temporal anomalies. [22] uses cell phone networks to record movements and interaction patterns of the population. Anomalies in the streaming of data produced by the cell phone network support a crisis management system. [36] proposes a solution that concerns drinking water surveillance and includes sensors and algorithms that detect anomalies in the properties of drinking water.

In our approach, anomaly detection is applied to improve the trust in the critical situation described by macro-events generated by the aggregation of heterogeneous input sources. This approach allows coping with the (spatial, temporal) uncertainty which is inevitably lying in the micro-events, thus also affecting the events at the macro level. To the authors' knowledge no works apply anomaly detection to such scope.

## 8 Conclusions

This paper presented an approach for critical situation detection that is able to integrate heterogeneous sources and it can possibly scale to any input source, thanks to the definition of an event model which exploits an ontology. The solution devised is an event processor that combines i) Complex Event Processing for the detection and composition of the events that describe the critical situation, and ii) anomaly detection to improve trustworthiness of the events reducing the uncertainty introduced by human sources.

A prototype of the proposed solution has been developed and exercised in the context of the Secure! project. The conducted experimental validation activities show the feasibility of the approach for on-line critical situation detection and its capabilities in detecting anomalous events that could impair the accuracy of the critical detection process.

Two families of tests were implemented. First, tests of the event processor and of the individual component Event Trust Analysis were performed to explore their performance. Second, the event processor was tested in a larger setting, where it is part of a crisis management system. Tests results showed that the event processor performs adequately for its intended purpose, and that the algorithms implemented for anomaly detection are adequate for the considered scenarios. In particular, we exercised the event processor in a world heritage protection scenario. This experience will be a relevant input for the heritage protection platform that is targeted in the starting project H2020-DRS-700191-STORM [60], where the two companies Engineering Ingegneria Informatica and Resiltech, co-authors of this paper, are involved. Amongst its objectives, STORM will explore heritage protection relying on several information sources as intra-fluorescent and wireless acoustic sensors, LiDAR (Light Detection and Ranging) sensors, UAVs (Unmanned Aerial Vehicles), and crowdsourcing. The STORM project will provide the ground to further consolidate the event processor i) promoting its execution in novel scenarios, and ii) showing its flexibility to different heterogeneous data sources and targets.

Beyond its use in crisis management systems, the event processor and the event trust analysis may be also adopted in other contexts, whenever multiple heterogeneous events are collected that can be classified and aggregated satisfying specific rules according to a given ontology, event model and correlation rules. This can result useful in several cases, roughly corresponding to the domains where the demand of complex event processing is increasing. Possible examples are [63] i) intrusion and failure detection to protect from attacks or failures [57], [58], ii) environmental monitoring applications which process data coming from multiple sensors, iii) traffic management and V2I (Vehicles to Infrastructure) communication.

## Acknowledgment

This work was partially supported by the research projects POR-CREO 2007-2013 "Secure!" funded by Regione Toscana, PRIN n. 20103P34XC "TENACE-Protecting National Critical Infrastructures from Cyber Threats" funded by the Italian Ministry of Education, University and Research, and the H2020-DRS-700191-STORM "Safeguarding Cultural Heritage through Technical and Organisational Resources Management" funded by the European Community.

## REFERENCES


[1] Christian Dietrich and Patryk Pawlak, “Crowd-sourcing - crisis response in the digital age”, European Union Institute for Security Studies, No 39 November 2013

[2] Raghu K. Ganti, Fan Ye, and Hui Lei, “Mobile Crowdsensing: Current State and Future Challenges”, IEEE Communications Magazine, Vol. 49, no. 11, pp. 32-39, 2011

[3] D. B. Beringer, and P. A. Hancock, “Exploring situational awareness: A review and the effects of stress on rectilinear normalisation”, in Proceedings of the Fifth International Symposium on Aviation Psycology (Vol2), pp. 646-651, 1989.

[4] Tomcat manager, [online] https://tomcat.apache.org/tomcat-7.0-doc/manager-howto.html

[5] M.L. Itria, A. Daidone, A. Ceccarelli, “A Complex Event Processing Approach for Crisis-Management Systems”, BIG4CIP Workshop at EDCC2014 (*informal article at* http://arxiv.org/abs/1404.7551), 2014.

[6] Secure! Project (2013-2015), Regione Toscana POR-CREO 2007-2013, http://secure.eng.it/.

[7] Dave Yates, Scott Paquette, Emergency knowledge management and social media technologies: A case study of the 2010 Haitian earthquake, International Journal of Information Management, Volume 31, Issue 1, February 2011, Pages 6-13, ISSN 0268-4012.

[8] O. Etzion and P. Niblett, “Event Processing in Action”, MANNING Greenwich (74° w. long.), 2011, pp.68-73.

[9] Gao, Huiji, Geoffrey Barbier, and Rebecca Goolsby. "Harnessing the crowdsourcing power of social media for disaster relief." IEEE Intelligent Systems 3 (2011): 10-14.

[10] B.Viswanath, M.A. Bashir, M. Crovella, S. Guha, K.P. Gummadi, B. Krishnamurthy, A. Mislove, “Towards Detecting Anomalous User Behavior in Online Social Networks”, 23th USENIX Simposium, August 2014.

[11] Thom, Dennis, et al. "Spatiotemporal anomaly detection through visual analysis of geolocated twitter messages." Pacific visualization symposium (PacificVis), 2012 IEEE. IEEE, 2012.

[12] Secure! Project, Events Taxonomy, 2013, secure.eng.it/ontologySecure/category.owl

[13] T. Zoppi et al., Presenting the Proper Data to the Crisis Management Operator: A Relevance Labelling Strategy, *in press at HASE 2016, 7-9 January 2016 (see supplemental material)*.

[14] Kaufman, L. and Rousseeuw P.J. (1987). Clustering by means of medoids, in: Y. Dodge (ed.), Statistical Data Analysis Based on the $L_1$-norm and Related Methods. Elsevier, North-Holland, pp. 405-416.

[15] R Core Team (2014). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

[16] RStudio, http://www.rstudio.com/.

[17] Maechler, M., Rousseeuw, P., Struyf, A., Hubert, M., Hornik, K.(2014). cluster: Cluster Analysis Basics and Extensions. R package version 1.15.2.

[18] H. Wickham. ggplot2: elegant graphics for data analysis. Springer New York, 2009.

[19] David Kahle and Hadley Wickham (2013). ggmap: A package for spatial visualization with Google Maps and OpenStreetMap. R package version 2.3. http://CRAN.R-project.org/package=ggmap

[20] Karline Soetaert (2014). plot3D: Plotting multi-dimensional data. R package version 1.0-1. http://CRAN.R-project.org/package=plot3D

[21] Daniel Adler, Duncan Murdoch and others (2014). rgl: 3D visualization device system (OpenGL). R package version 0.93.996. http://CRAN.R-project.org/package=rgl

[22] Pawling, Alec, et al. "Anomaly detection in streaming sensor data." Intelligent Techniques for Warehousing and Mining Sensor Network Data (2008): 99-117.

[23] Chris Peltz, "Web Services Orchestration and Choreography," Computer, vol. 36, no. 10, pp. 46-52, October, 2003.

[24] Höhle, M. (2007). *Surveillance*: An R package for the monitoring of infectious diseases. Computational Statistics, 22(4), 571-582

[25] Auslander, Bryan, Kalyan Moy Gupta, and David W. Aha. "A comparative evaluation of anomaly detection algorithms for maritime video surveillance." SPIE Defense, Security, and Sensing. International Society for Optics and Photonics, 2011.

[26] V. Vianello, V. Gulisano, R Jmenez-Pers, M. Platino-Martinez, R. Torres, R. Díaz and E. Prieto, “A Scalable SIEM Correlation Engine and its Application to the Olympic Games IT Infrastructure” in Proceedings of International Conference on Availability, Reliability and Security, 2013.

[27] Lorincz, Konrad, et al. "Sensor networks for emergency response: challenges and opportunities", Pervasive Computing, IEEE 3.4 (2004): 16-23.

[28] Andrea Bondavalli, Andrea Ceccarelli, Lorenzo Falai, Michele Vadursi: Foundations of Measurement Theory Applied to the Evaluation of Dependability Attributes. DSN 2007: 522-533

[29] Paraiso, F.; Hermosillo, G.; Rouvoy, R.; Merle, P.; Seinturier, L., "A Middleware Platform to Federate Complex Event Processing," Enterprise Distributed Object Computing Conference (EDOC), 2012 IEEE 16th International , vol., no., pp.113,122, 10-14 Sept. 2012.

[30] Esper Team and EsperTech Inc. “Esper Reference version 4.9.0”, 2012 http://esper.codehaus.org [last accessed 24th November 2014].

[31] Chandola, Varun, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey." ACM Computing Surveys (CSUR) 41.3 (2009):

[32] Barthe-Delanoë et al. “Event-driven agility of crisis management collaborative processes”, In Proceedings of the 9th International ISCRAM Conference, 2012.

[33] Vassilios Vescoukis, Nikolaos Doulamis, Sofia Karagiorgou: A Service Oriented Architecture for Decision Support Systems in environmental Crisis Management. In Future Generation Computer Systems, Vol. 28, March 2012, pp 593–604.

[34] Fleischer, J., et al. "An integration platform for heterogeneous sensor systems in GITEWS–Tsunami Service Bus." Natural Hazards and Earth System Science 10.6 (2010): 1239-1252.

[35] Meier, Patrick. "New information technologies and their impact on the humanitarian sector." International review of the Red Cross 93.884 (2011): 1239-1263.

[36] Eriksson, Mats, et al. "Event detection in crisis management systems." Procedia Chemistry. Vol. 1. No. 1. Elsevier, 2009.

[37] Bénaben, Frédérick, et al. "A metamodel and its ontology to guide crisis characterization and its collaborative management", Proc. of the 5th International Conference on Information Systems for Crisis Response and Management (ISCRAM), Washington, DC, USA, May. 2008.

[38] Muhammad Imran, Carlos Castillo, Fernando Diaz, and Sarah Vieweg, “Processing Social Media Messages in Mass Emergency: A Survey”, ACM Comput. Surv. 47, 4, Article 67, 2015.

[39] Siti Hajar Othman, Ghassan Beydoun, Vijayan Sugumaran, “Development and validation of a Disaster Management Metamodel (DMM)”, Information Processing & Management, Volume 50, Issue 2, 2014, pp.235-271.

[40] MySQL admin, [online] http://dev.mysql.com/doc/refman/5.7/en/mysqladmin.html

[41] Faulkner, Matthew, et al. "Community sense and response systems: Your phone as quake detector", Communications of the ACM 57.7 (2014): 66-75.

[42] L. Liu, D. Webster, J. Xu, and K. Wu, “Enabling dynamic workflow for disaster monitoring and relief through service-oriented sensor networks”, In CHINACOM, Beijing, China, Aug. 2010.

[43] Netten, Niels, and Maarten van Someren, "Automated support for dynamic information distribution in incident management", Proceedings of the 3 International ISCRAM Conference, 2006.

[44] DBPedia, [online] http://wiki.dbpedia.org/

[45] Amerini, Irene, et al. "Media trustworthiness verification and event assessment through an integrated framework: a case-study", Multimedia Tools and Applications (2016): 1-16.

[46] https://github.com/AndreaCeccarelli/Event-processor-dataset

[47] Liu, Shuangyan, Duncan Shaw, and Christopher Brewster, "Ontologies for crisis management: a review of state of the art in ontology design and usability", Proceedings of the Information Systems for Crisis Response and Management conference (ISCRAM), 2013.

[48] de la Asunción, Marc, et al. "SIADEX: An interactive knowledge-based planner for decision support in forest fire fighting", AI Communications 18.4 (2005): 257-268.

[49] D. Fawzy and Y. Sahin, "RT-HRLE: A system design for real-time hazards reporting and loss estimation using wireless sensors", In Intern. Conf. on Education and Management Technology (ICEMT), Cairo, Egypt, Nov. 2010.

[50] Babitski, G., Probst, F., Hoffmann, J., Oberle, D., "Ontology design for information integration in disaster management", Proceedings of the 4th International Workshop on Applications of Semantic Technologies (AST), 2009.

[51] Chen-Huei, C., Zahedi, F.M., Huimin, Z., "Ontology for Developing Web Sites for Natural Disaster Management: Methodology and Implementation", Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on, 41, 1, 50-62, 2009.

[52] Canadian Disaster Database, [online] http://www.publicsafety.gc.ca/cnt/rsrcs/cndn-dsstr-dtbs/index-eng.aspx

[53] Australian AG Disasters Database, [online] https://www.emknowledge.gov.au/disaster-information/

[54] EM-DAT, The International Disaster Database, [online] http://www.emdat.be/classification

[55] Humanitarian Exchange Language (HXL), [online] http://hxlstandard.org/

[56] Il Tirreno, "Nei Musei solo otto turisti su cento" (in Italian: only 8 tourists out of 100 enter museums), 26 April 2009, [online] http://rassegnastampa.unipi.it/rassegna/archivio/2009/04/27SIB1024.PDF

[57] Alan S. Willsky, "A survey of design methods for failure detection in dynamic systems", Automatica, 12(6):601–611, November 1976.

[58] Imran, Muhammad, et al. "Aidr: Artificial intelligence for disaster response", Proceedings of the companion publication of the 23rd International Conference on World wide web. 2014.

[59] Sokat, Kezban Yagci, et al. "Capturing real-time data in disaster response logistics", No. 14-05. Working Paper, 2014.

[60] H2020-DRS-700191-STORM "Safeguarding Cultural Heritage through Technical and Organisational Resources Management".

[61] J. Rogstadius, et al. "CrisisTracker: Crowdsourced social media curation for disaster awareness", in IBM Journal of Research and Development, vol. 57, no. 5, pp. 4:1-4:13, Sept.-Oct. 2013.

[62] MacEachren, Alan M., et al. "Senseplace2: Geotwitter analytics support for situational awareness", Visual Analytics Science and Technology (VAST), 2011 IEEE Conference on. IEEE, 2011.

[63] Cugola, Gianpaolo, and Alessandro Margara, "Processing flows of information: From data stream to complex event processing", ACM Computing Surveys (CSUR) 44.3 (2012): 15.

[64] Shams, Farshad, Antonio Cerone, and Rocco De Nicola, "On Integrating Social and Sensor Networks for Emergency Management", Software Engineering and Formal Methods. Springer Berlin Heidelberg, 2015. 145-160.

[65] Power, Robert, et al. "Emergency situation awareness: Twitter case studies", Information Systems for Crisis Response and Management in Mediterranean Countries. Springer International Publishing, 2014. 218-231.

[66] Alexandra Olteanu, Carlos Castillo, Fernando Diaz, and Sarah Vieweg, "CrisisLex: A lexicon for collecting and filtering microblogged communications in crises", In Proc. of ICWSM, 2014.

[67] Muhammad Imran et al. "Extracting information nuggets from disaster-related messages in social media", In Proc. of ISCRAM, 2013.

[68] Leavitt, Alex, and Joshua A. Clark, "Upvoting hurricane Sandy: event-based news production processes on a social news site", Proceedings of the SIGCHI conference on human factors in computing systems. ACM, 2014.

[69] Caragea, Cornelia, et al. "Classifying text messages for the Haiti earthquake", In Proc. of ISCRAM, 2011.

[70] Panagiotis Metaxas and Eni Mustafaraj, "The rise and the fall of a citizen reporter", In Proc. of WebSci, 2013.

[71] Aslam, Javed A., et al. "TREC 2014 Temporal Summarization Track Overview" TREC, 2013.

[72] Shou, Lidan, et al. "Sumblr: continuous summarization of evolving tweet streams", Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval. ACM, 2013.

[73] Abel, Fabian, et al. "Semantics+filtering+search=twitcident. exploring information in social web streams", Proceedings of the 23rd ACM conference on Hypertext and social media. ACM, 2012.

[74] Imran, Muhammad, et al., "Coordinating human and machine intelligence to classify microblog communications in crises", Proc. of ISCRAM, 2014.

**Massimiliano Leone Itria** computer engineer, graduated at University of Pisa, started his research experience in 2010 at the Consiglio Nazionale delle Ricerche (CNR) in Pisa where he worked as software designer for resilient systems and he dealt with Dependability and Performance Analysis of connected systems. He also had experience in safety critical systems and in particular in railway systems. Since march 2013 he has been working in ResilTech and he is currently involved in several research European projects giving contributions for designing innovative network monitoring systems, anomaly detection in System of Systems, and Situational Awareness systems aimed to the Cultural Heritage protection. He is an expert of Information Fusion techniques, Complex Event Processing technologies, Natural Language Processing, Emergency Management Systems and Risk Analysis.

**Melinda Kocsis-Magyar** received her MSc degree in Software Engineering in 2006 from the Budapest University of Technology and Economics. She participated in EU FP7 academic research projects including SAFEDMI, HIDENETS, RESIST and AMBER. Her main research field was model-based dependability analysis of safety critical systems and verification and validation of dependability mechanisms. She works for a process control company as a system engineer where she prepared technical documentations for the EU project "Plovdiv-Svilengrad Railway Electrification and Upgrading of Corridors IV and IX" (including RAMS analysis, design and maintenance documentation) and a product development project (RAMS analysis, requirement management and design document verification responsibilities as a senior engineer).

**Andrea Ceccarelli** received the PhD degree in Informatics and Automation Engineering from the University of Florence, Italy, in 2012. Since then, he is a Research Associate at the Department of Mathematics and Informatics, University of Florence. His research activity is mainly focused on the design and the experimental evaluation of complex dependable and secure systems and systems-of-systems. He has been involved in European and National funded projects since 2006, and he is presently involved in the European funded projects FP7-PEOPLE-2012-IAPP CECRIS, FP7-PEOPLE-2013-IRSES DEVASSES, and the JPI URBAN EUROPE IRENE. His scientific activities originated more than 60 papers appeared in International Conferences, Workshops and Journals and he served in the Program Committee of several International Conferences including DSN, SRDS.

**Paolo Lollini** received the PhD degree in computer science from the University of Florence, Italy, in 2005. Since 2006 he worked as research associate at the same University, and he is currently assistant professor at the Mathematics and Computer Science Dept. He has been continuously participating in European and National funded projects since 2002 up to present, including the recently concluded projects ICT-FP7-610535 AMADEOS, ARTEMIS-JU-333053 CONCERTO and PRIN-20103P34XC TENACE (National), and he is currently participating to the PIRSES-GA-2013-612569 DEVASSES project. He was a member of the program committee of important conferences in the area of dependable systems, including DSN,

HASE, LADC, SRDS, and currently EDCC. His current research interests include the stochastic modelling and evaluation of performability and resiliency attributes of large-scale critical infrastructures and systems-of-systems, with reference to a variety of application fields including railway, mobile telecommunications, and electric power systems.

**Gabriele GIUNTA** obtained a degree in Computer Science, at the Engineering Department of the University of Palermo (Italy). He is head of the "Smart Transport and Infrastructure" Unit within the R&D Laboratory at ENGINEERING ING. INF. S.p.A. He has coordinated and participated in several research projects, in the field of Smart Transport and Critical Infrastructure Protection. His relevant expertise includes Software Architecture Design, Software Engineering, System Integration and Service Oriented Computing. He has been co-author of several scientific papers in International Conferences and Journals.

**Andrea Bondavalli** is a Professor of Computer Science at the University of Firenze. Previously he has been a researcher and a senior researcher of the Italian National Research Council, working at the CNUCE Institute in Pisa. His research activity is focused on Dependability and Resilience of critical systems and infrastructures. In particular he has been working on safety, security, fault tolerance, evaluation of attributes such as reliability, availability and performability. His scientific activities have originated more than 220 papers appeared in international Journals and Conferences. Andrea Bondavalli led various national and European projects and participates to (and has been chairing) the program committee in several International Conferences in the field. He is the chair of the Steering Committees of IEEE SRDS and a member of the editorial board of the International Journal of Critical Computer-Based Systems. Andrea Bondavalli is a member of the IEEE, the IFIP W.G. 10.4 Working Group on "Dependable Computing and Fault-Tolerance".