On-line integration and reasoning of multi-sensor data to enhance infrastructure surveillance

Francesco Flammini[12], Andrea Gaglione[2], Nicola Mazzocca[2], Vincenzo Moscato[2] and Concetta Pragliola[1]

[1]Ansaldo STS Italy
Innovation Unit
Via Argine 425, Naples, Italy
{francesco.flammini, concetta.pragliola}@ansaldo-sts.com

[2]University of Naples Federico II
Department of Computer and Systems Engineering
Via Claudio, 21, Naples, Italy
{frflammi, andrea.gaglione, nicola.mazzocca, vmoscato}@unina.it

Abstract: Modern security systems employed in infrastructure surveillance applications include a set of different sensing technologies integrated by appropriate management systems. Such systems are still highly dependent from human operators for supervision and intervention. One of the challenging goals of the research community in this field is the automatic detection of both natural and malicious threat scenarios. In this paper we describe a framework for correlating events detected by sensor networks to provide early warning and decision support capabilities. To this aim, we propose an overall system architecture for the integration of the DETECT and SeNsIM frameworks: DETECT (Decision Triggering Event Composer & Tracker) is a new system which is able to recognize complex events by a model-based correlation of primitive events; SeNsIM (Sensor Networks Integration and Management) is a middleware for the integration of heterogeneous sensor networks. The paper also provides example application scenarios in the railway domain.

Keywords: Sensor Networks, Smart Surveillance, Critical Infrastructure Protection, Sensor Data Fusion, Soft Computing

1. Introduction

The best way to face threats is to stop them before they can cause catastrophic consequences. Unfortunately, visual surveillance of video streams and sensor alarms provided by current security systems does not allow operators for a satisfactory situational awareness when the sequence of events is large, heterogeneous, geographically distributed and rapidly evolving. Therefore, operators are hardly able to recognize sequences of events which are indicative of possible threats due to their limited alert threshold and knowledge base. Furthermore, operators can be unable to guide and coordinate alarm responses or emergency interventions if they are not precisely aware of what is happening or has happened. In order to cope with these issues, early warning and decision support systems can be adopted.

The aim of this work is to propose the architecture for a decision support and early warning system used to effectively face security threats (e.g. terrorist attacks) based on heterogeneous distributed sensor networks. Therefore, this work locates at the third stage (i.e. "Indications and warning") of the Critical Infrastructure Protection life-cycle (see Figure 1).

In particular, smart-sensors used in Wireless Sensor Networks (WSN) feature several advantages when applied to critical infrastructure surveillance [23], as they are:

- Cheap, and this allows for fine grained and highly redundant configurations;
- Resilient, due to their fault-tolerant mesh topology;
- Power autonomous, due to the possibility of battery and photovoltaic energy supplies;
- Easily installable, due to their wireless nature and auto-adapting multi-hop routing;
- Intelligent, due to the on-board processor and operating systems which allow for some data elaborations being performed locally.

All these features support the use of WSN in highly distributed monitoring applications in critical environments. The example applications we will refer to in this paper are in the domain of rail-based transport infrastructures, which need to be protected against external threats which can be natural (fire, flooding, landslide, etc.) or human-made malicious (sabotage, terrorism, etc.).

Examples of useful sensors in this domain are listed in the following: smoke and heat – useful for fire detection; moisture and water – useful for flooding detection; pressure – useful for explosion detection; movement detection (accelerometer or satellite based shifting measurement) – useful for theft detection or structural integrity checks; gas and explosive – useful for chemical or bombing attack detection; vibration and sound – useful for earthquake or crash detection. WSN could also be used for video surveillance and on-board intelligent video-analysis, as reported in [19].

The heterogeneity of network topologies and measured data requires integration and analysis at different levels (see Figure 2).

As first, the monitoring of wide geographical areas and the diffusion of sensor networks managed by different middlewares have highlighted the research problem of the integrated management of data coming from the various networks. Unfortunately such information is not available in a unique container, but in distributed repositories and the major challenge lies in the heterogeneity of repositories which complicates data management and retrieval processes.

This issue is addressed by the SeNsIM framework [5], as described in Section 3.

Secondly, there is the need for an on-line reasoning about the events captured by sensor nodes, in order to early detect and properly manage security threats. The availability of possibly redundant data allows for the correlation of basic events in order to increase the probability of detection, decrease the false alarm rate, warn the operators about suspect situations, and even automatically trigger adequate countermeasures by the Security Management System (SMS). This issue is addressed by the DETECT framework [10], as described in Section 4.

The rest of the paper is organized as follows: Section 5 discusses about the SeNsIM and DETECT software integration; Section 6 introduces example railway security applications; Section 7 draws conclusions and hints about future developments.

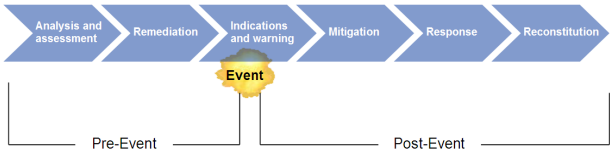

Figure 1. Critical Infrastructure Protection life-cycle.

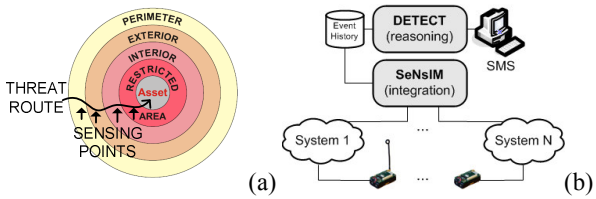

Figure 2. (a) Distributed sensing in physical security; (b) Overall monitoring architecture

2. Related works

In this section we present some integration platforms for heterogeneous sensor systems as well as most well known techniques for composite event detection.

2.1. Integration platform for sensor systems

The need for guaranteeing the interoperability among several monitoring systems have highlighted the problem of the integrated management of data coming from different networks. Interesting researches related to integration techniques for heterogeneous sensor networks have taken place, but to date only a few architecture have been proposed.

In [2] Ahn and Chong propose an intelligent bridge as an interoperable architecture for messages exchange between heterogeneous wireless sensor networks. They define a general messages exchange mechanism that uses XML as message style and SOAP as transmission protocol. They also conducted a case study based on two WSNs with different network protocols (ZigBee and Bluetooth).

Hourglass [24] provides an Internet-based infrastructure for connecting sensor networks to applications. It offers topic-based discovery and data-processing services, but focuses on maintaining quality of service of data streams in the presence of disconnections.

GSN (Global Sensor Networks) [1] facilitates the flexible integration and discovery of sensor networks and hides arbitrary stream data sources behind its virtual sensor abstraction. It enables the user to specify XML-based deployment descriptors in combination with the possibility to integrate sensor network data through plain SQL queries over local and remote sensor data sources.

The *ESP* framework [21] enables sensor systems to be queried without having to deal with the low-level implementation of specific access methods. It provides a mechanism to describe and model sensor systems using *ESPml*, an XML-based language, by which information regarding the sensor deployment can be specified. Furthermore, the ESP architecture is based on web services as interoperability platform.

Eventually, *IrisNet* [15] can be considered a hybrid approach to integration. It is a web infrastructure for easy deployment of wide-area sensing services. The architecture consists of *sensing agents* (SA) which collect and pre-process sensor data and *organizing agents* (OA) which store sensor data in a hierarchical, distributed XML database that supports XPath queries.

Most of these approach are still in a design phase since they lack a real implementation. Some try to define a common exchange mechanism among different sensor systems in order to facilitate the integration, but often they are strongly related to the adopted standards and technologies. Others are based on new technologies (i.e. web services for ESP), but the impact on performances is not discussed, yet. IrisNet does not consider constraints and characteristics of WSNs: it has introduced the notion of wide-area "sensor webs", such as those comprising Internet connected, widely-dispersed PC-class nodes with powerful CPUs that can process rich sensor data sources.

In contrast to the examined approaches, the SeNsIM system provides a general-purpose infrastructure for sensor systems integration, and a more general way to express queries by introducing a query visual language. It is a complete data retrieval and management platform with a simple user interface by which is possible to execute queries.

2.2. Composite event detection

Composite event detection plays an important role in the active database research community, which has long been investigating the application of Event Condition Action (ECA) paradigm in the context of using triggers, generally associated with update, insert or delete operations. In HiPAC [9] active database project an event algebra was firstly defined.

Our approach for composite event detection follows the semantics of the Snoop [7] event algebra. Snoop has been developed at the University of Florida and its concepts have been implemented in a prototype called Sentinel ([8], [17]). Event trees are used for each composite event and these are merged to form an event graph for detecting a set of composite events. An important aspect of this work lies in the notion of *parameter contexts*, which augment the semantics of composite events for computing their parameters (parameters indicate "component events"). CEDMOS [6]

refers to the Snoop model in order to encompass heterogeneity problems which often appear under the heading of sensor fusion. In [3] the implementation of an event detection engine that detects composite events specified by expressions of an illustrative sublanguage of the Snoop event algebra is presented. The engine ha been implemented as a Web Service, so it can also be used by other services and frameworks if the markup for the communication of results is respected.

Different approaches for composite event detection are taken in Ode [14] and Samos [13]. Ode uses an extended Finite Automata for composite event detection while Samos defines a mechanism based on Petri Nets for modeling and detection of composite events for an Object Oriented Data-Base Management System (OODBMS).

DETECT transfers to the physical security the concept of Intrusion Detection System (IDS) which is nowadays widespread in computer (or "logical") security, also borrowing the principles of Anomaly Detection, which is applied when an attack pattern is known a priori, and Misuse Detection, indicating the possibility of detecting unknown attacks by observing a significant statistical deviation from the normality [16]. The latter aspect is strictly related to the field of Artificial Intelligence and related classification methods.

Intelligent video-surveillance exploits Artificial Vision algorithms in order to automatically track object movements in the scene, detecting several type of events, including virtual line crossing, unattended objects, aggressions, etc. [22].

3. The SeNsIM framework

The SeNsIM system is an integration platform for heterogeneous sensor systems. It is not just a middleware for sensor networks, but a more general software architecture which makes possible the deployment of applications based on multiple sensor systems/networks and allows a generic user or an application to easily access data sensed by a network.

SeNsIM provides a unique interface for local networks that allow a generic user to express queries by means of an intuitive query visual language. It was conceived with the aim to bridge the gap between heterogeneous sensor systems and to provide a generic user/application with a unique way to manage, query and interact with them. Nowadays there is a tremendous heterogeneity in the logic for interfacing and collecting data from sensor systems. In most of them an application can directly access to sensor hardware by means of opportune drivers. In many different scenarios, as in WSN, an operating system between the hardware platform and the application allows an easier use of the available sensing functions and often a further middleware layer between the operating system and the user/application provides an easy way to access sensor data. The major issue of an integrating system lies in the heterogeneity of the hardware to sense data and of the repositories, these make data management and retrieval process a hard task to achieve.

Furthermore, sensor data may be differently structured according to the specific representation of different sensor systems. In order to face such problems SeNsIM defines:

- an *architectural model* able to support in an efficient way the management of data even when sensed by different networks;
- a *data model* capable of representing in a unique format both sensor data and sensor systems.

SeNsIM architectural model has been designed by exploiting the *wrapper-mediator* paradigm, a well-known technique to integrate data from heterogeneous source [26], in which the *mediator* component accesses different data sources by means of ad hoc connectors (*wrappers*). In SeNsIM, each wrapper explores and monitors the local sensor network and sends to the mediator an appropriate description of the related information according to the common data model. On the other hand the mediator organizes such information and keeps a unique view of all systems in order to satisfy user or application queries.
SeNsIM architectural model is made of 4 logical layers (see Figure 3):

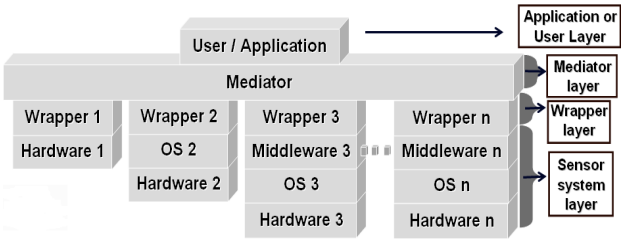

Figure 3. SeNsIM architectural model

1. The **Application** or **User Layer** allows a generic user to submit queries and elaborate the retrieved data; a generic application can also access sensor data through system API. The system provides support for monitoring queries which return the corresponding responses in real-time as well as for event queries. Many context aware applications need to trigger adequate actions/ countermeasures after that some events have been generated by sensor systems.
2. The **Mediator Layer** aims to classify networks features as well as to format and forward queries to specific wrappers; a DBMS is used to store data related to networks with their sensors, user queries and related results.
3. The **Wrapper Layer** aims to extract and manage information about the underlying network and its sensors; at this layer queries from mediator are received and executed on the local system by using its API and the local query language. A DBMS in each wrapper component is kept to for storing network/sensors information according to the data model and local queries with related results.
4. The **Sensor system Layer** aims to extract network features and carry out the retrieval process.

SeNsIM data model is able to represent a *sensor node* as well as the whole *network*. According to the model a sensor node is an object characterized by a tuple of information which

combine both structural and behavioral features. Our model is able to represent sensor global information (type, producer, description, etc...) as well as the variables that a sensor can measure (temperature, light, humidity, etc...), predicates that a sensor can calculate (e.g. temperature greater than a threshold) and sensor operating state/mode (i.e. continuous monitoring, event-driven monitoring). Further, a network object has to include global information such as type of sensor system, middleware (if present), supported sensor board as well as information related to sensor components (list of sensors, possible list of clusters, topology matrix). Moreover it is possible to specify network global predicates (e.g. average temperature of the network).

XML has been used to represent the data model, since it provides platform independence, interoperability and can be easily parsed [25]. XML-based descriptors let have a unifying grammar by which systems can describe their abilities and define a standard language protocol with which the different entities in the framework can communicate. In SeNsIM XML descriptors are exchanged between mediator and wrappers components. The communication between wrappers and mediator is carried out by using TCP and UDP sockets according to the SeNsIM proprietary communication protocol. The reference architecture of the SeNsIM system is shown in Figure 4.

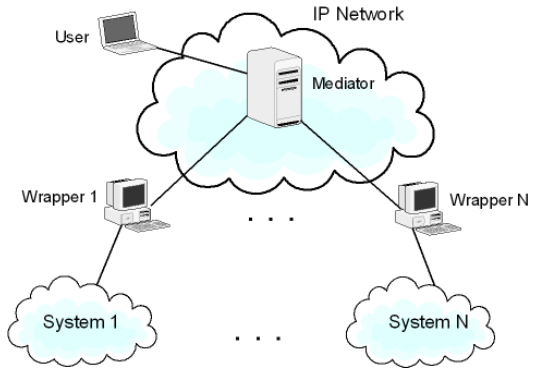

Figure 4. SeNsIM architecture

4. The DETECT framework

The basic assumption behind the DETECT framework is that threats can be detected by predicting the set of basic events (i.e. the patterns) which constitute their "signature". For instance, Figure 2a shows the multi-layered asset protection provided by modern security systems. In each layer a set of sensors (i.e. video, motion, temperature, vibration, sound, smoke, etc.) are installed. Threat scenarios must be precisely identified during Vulnerability Assessment and Risk Analysis. DETECT operates by performing a model-based logical, spatial and temporal correlation of basic events detected by different sensor subsystems, in order to recognize sequence of events which indicate the likelihood of threats. DETECT is based on a real-time detection engine which implements the concepts of data fusion and cognitive reasoning by means of soft computing approaches. The framework can be interfaced with or integrated in existing SMS (Security Management Software). It can serve as an early warning tool or even to automatically trigger adequate countermeasures for emergency/crisis management. As such, it may allow for a quick, focused and automatic response to emergencies, though manual confirmation of detected alarms remains an option. In fact, human management of critical situations, possibly involving many simultaneous events, is a very delicate task, which can be error prone as well as subject to forced inhibition. Used as a warning system, it can alert the operators about the likelihood and nature of the threat; used as an autonomous reasoning engine, it can activate responsive actions, including audio and visual alarms, unblock of turnstiles, air conditioned flow inversion, activation of sprinkles, emergency calls to first responders, etc. Furthermore, the correlation among basic events detected by diverse redundant sensors can also be employed to lower the false alarm rate of the security system, thus improving the overall reliability of the security system.

Threats are described in DETECT using a specific *Event Description Language (EDL)* and stored in a *Scenario Repository*. Starting from the Scenario Repository, one or more detection models are automatically generated using a suitable formalism (e.g. Event Graphs, Bayesian Networks, Neural Networks, etc.). In the operational phase, a model manager macro-module has the responsibility of performing queries on the Event History database for the real-time feeding of detection model according to predetermined policies. When a composite event is recognized, the output of DETECT consists of: the identifier(s) of the detected/suspected scenario(s); an alarm level, associated to scenario evolution (used as a progress indicator); a likelihood of attack, expressed in terms of probability (used as a threshold in heuristic detection). A high level architecture of the framework is depicted in Figure 5.

Together with a sensor network integration framework, DETECT can perform fusion and reasoning of data generated by smart wireless sensors. To this aim, DETECT has to be integrated on top of SeNsIM (Sensor Networks Integration and Management), as depicted in Figure 2b.

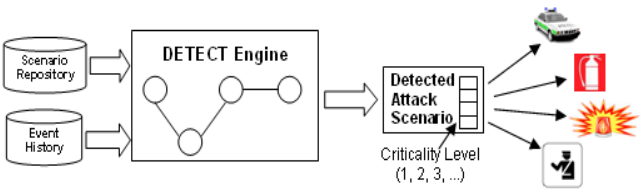

Figure 5. The DETECT framework

2.3. Event Description and representation

The Detection Engine needs to recognize combination of events, bound each other with appropriate operators in order to form composite events of any complexity. Generally speaking, an event is a happening that occurs in the system, at some location and at some point in time. In our context, events are related to sensor data variables (i.e. variable x greater than a fixed threshold, variable y in a fixed range, etc.). Events are classified as *primitive events* and *composite events.*

A primitive event is a condition on a specific sensor which is associated some parameters (i.e. event identifier, time of occurrence, etc). Since the message transportation time is not instantaneous, the event occurrence time can be different

from the registration time. Several research works have addressed the issue of clock synchronization in distributed systems. Here we assume that a proper solution (e.g. time shifting) has been adopted at a lower level.

A composite event is a combination of primitive events by means of proper operators. The EDL of DETECT is derived from the Snoop event algebra [7]. Every composite event instance is a triple:

< IDec, parcont, te >

where:

- **IDec** is the composite event identifier;
- **parcont** is the parameter context, stating which occurrences of primitive events need to be considered during the composite event detection (as described below);
- **te** is the temporal value related to the occurrence of the composite event (corresponding to the tp of the last component event).

Each event is denoted by an *event expression*, whose complexity grows with the number of involved events. Given the expressions E_1, E_2, …, E_n, every application on them through any operator is still an expression. We use both logical and temporal operators, for a formal specification of their semantics, the reader can refer to [6]. Event expression are represented by *event tree*, where primitive events are at the leaves, while internal nodes represent EDL operators. Figure 6 shows an example event tree representing a composite event.

The engine also performs time correlation on events by defining and exploiting temporal constraints. In fact, logic correlation could loose meaningfulness when the time interval between correlated events exceeds a certain threshold. *Temporal constraints* can be defined on both primitive and composite events with the aim of defining a validity interval for the overall composite event. Such constraints can be added to any operator in the formal expression used for event description.

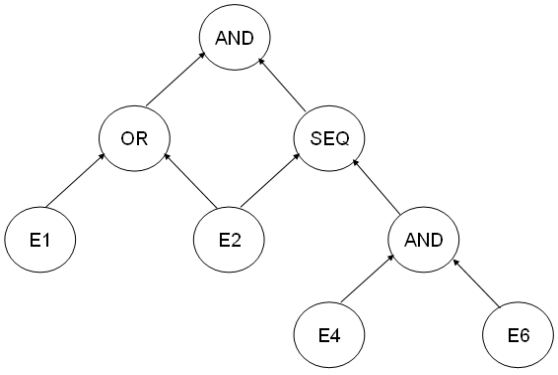

Figure 6. Event tree for composite event ((E1 OR E2) AND (E2 SEQ (E4 AND E6)))

For instance, let us assume that in the composite event E = (E_1 AND E_2) the time interval between the occurrence of primitive events E_1 and E_2 must be at most T. The formal expression is modified by adding the temporal constraint [T] as follows:

$$(E_1 \text{ AND } E_2)\,[T] = \textit{True}$$
$$\Leftrightarrow$$
$$\exists\, t_1 < t \mid (E_1(t) \wedge E_2(t_1) \vee E_1(t_1) \wedge E_2(t)) \wedge |t - t_1| \leq T$$

2.4. The software architecture

The framework is made up by the following main modules (see Figure 7):

- **Event History**, that is database containing the list of basic events detected by sensors or cameras, tagged with a set of relevant attributes including detection time, event type, sensor id, sensor type, sensor group, object id, etc. (some of which can be optional, e.g. "object id" is only needed when video-surveillance supports inter-camera object tracking). A specific external **Events Adaptor Module** aims to fill the Event History with the primitive events coming from a monitoring sensor network. The functionalities offered by such a module can be accomplished the SeNsIM system, as we will illustrate in the next section.
- **Detection Engine**, supporting both deterministic (e.g. *Event Trees*, *Event Graphs*) and heuristic (e.g. *Artificial Neural Networks*, *Bayesian Networks*) models, sharing the primary requirement of real-time solvability (which excludes e.g. *Petri Nets* from the list of candidate formalisms). For each **Detection Model** there is a **Model Feeder** which instantiates the inputs of the engine according to the nature of the models by critically performing proper queries and data filtering on the Event History (e.g. selecting sensor typologies and zones, excluding temporally distant events, etc.). At the moment, the detection engine is only based on the deterministic model of the event trees.
- **Model Solver**, that is the existing or specifically developed tool used to execute the model. It implements the logical assumptions to solve the Detection Model, based on the inputs coming from the Model Feeder, therefore it is the responsible module for the composite event detection. We have implemented our own Model Solver based on the event trees Detection Model.
- **Model Executor** (one for each model), which triggers the execution of the mode, once it has been instantiated, by activating the related solver. An execution is usually needed at each new event detection.
- **Model Updater** (one for each model), which is used for on-line modification of the model (e.g. update of a threshold parameter), without regenerating the whole model (whenever supported by the modeling formalism).
- **Output Manager** (single), which stores the output of the model(s) and/or passes it to the interface modules.

- **Scenario GUI (Graphical User Interface)**, used to draw attack scenarios using an intuitive formalism and a user-friendly interface. Once a scenario has been built, it will be converted in a XML document by the **XML File Generator** module and then indexed in the **Attack Scenario Repository**. In this way we are able to store permanently all scenario features in a formal way as well as to facilitate possible subsequent data processing by other applications. In the opposite way, when the user selects the attack scenario he wants to detect from the repository, the XML document which contains its description has to be re-converted in the detection model, which represents the composite event related to the scenario. This task is carried out by the **Model Generator** which recover the original graph and its parameter by parsing the related XML document.

DETECT makes it possible to associate different alarm levels to each composite event as well as to its component sub-event which have to be signaled by the detection engine. In this way the user can be aware of the attack scenarios since their first evolution steps. Alarms could be sent to existing SMS/SCADA systems in order to trigger adequate countermeasures. However an **Event Log** is kept to gather all information about detected events (detection time, alarm level, instances of component events involved in the composite event detection process).

5. Integration of SeNsIM and DETECT

The SeNsIM and DETECT frameworks need to be integrated in order to obtain an on-line reasoning about the events captured by different sensor systems. As mentioned above, the aim is to early detect and manage security threats against critical infrastructures. In this section we provide the description of the sub-components involved in the software integration of SeNsIM and DETECT.

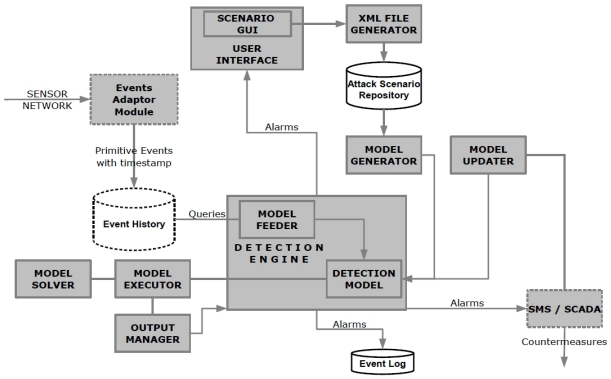

Figure 7. DETECT software architecture

During the query processing task of SeNsIM, user queries are first submitted by means of a *User Interface*; then, a specific module (*Query Builder*) is used to build a query. The user queries are finally processed by means of a *Query Processing* module which sends the query to the appropriate wrappers. The partial and global query results are then stored in a database named *Event History*. All the results are captured and managed by a *Results Handler*, which implements the interface with wrappers.

The Model Feeder is the DETECT component which performs periodic queries on the Event History to access primitive event occurrences. The Model Feeder instantiates the inputs of the Detection Engine according to the nature of the model(s).

Therefore, the integration is straightforward and mainly consists in the management of the Event History as a shared database, written by the mediator and read by the Model Feeder according to an appropriate concurrency protocol.

In Figure 8 we report software modules involved in the integration process between SeNsIM and DETECT. The figure also shows the modules of SeNsIM involved in the retrieval process, in particular:

- **Query Processing** is a macro-module, containing several sub-modules;

- **GUI (Graphical User Interface)** is used to edit DETECT scenarios using a graphical formalism translatable to EDL files. Moreover it allows the user to define SeNsIM queries for sensor data retrieval. User interaction is only needed in the configuration phase, to define attack scenarios and query parameters. According to the query strategy, both SeNsIM and DETECT can access data from the lower layers using either a cyclic or event driven retrieval process.

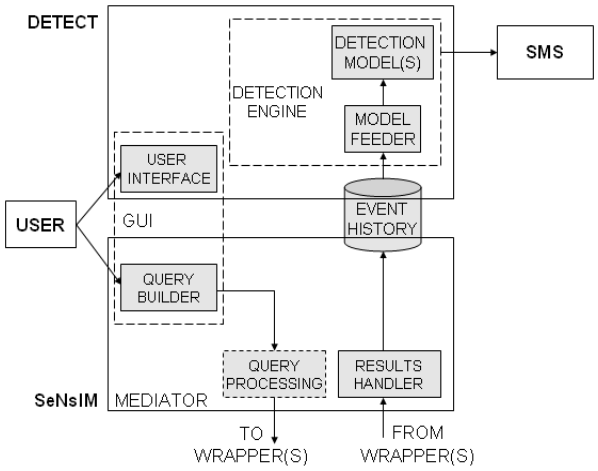

Figure 8. Query processing and software integration

6. Example application scenarios

In this section we report two example applications of the overall framework to case-studies related to rail-based transportation systems, which are attractive targets for thieves, vandals and terrorists. Several application scenarios can be thought exploiting the proposed architecture and several sensors (intrusion detection, track line break detection, on-track obstacle detection, etc.) and actuators

(e.g. alarms, public address, virtual or light signalling devices, etc.) could be installed to monitor system integrity against external threats and notify anomalies. In the following we describe how to detect some complex scenarios, including terrorist strategic attacks.

6.1 Explosive in a subway tunnel

The first attack scenario consists of an intrusion and drop of an explosive device in a subway tunnel. Let us suppose that the dynamic of the scenario follows the steps reported below:

1. The attacker stays on the platform for the time needed to prepare the attack, missing one or more trains.
2. The attacker goes down the tracks by crossing the limit of the platform and moves inside the tunnel portal.
3. The attacker drops the bag containing the explosive device inside the tunnel and leaves the station.

Obviously, it is possible to think of several variants of this scenario. For instance, only one between step 1 and step 2 could happen. Please note that the detection of step 1 (person not taking the train) would be very difficult to detect by a human operator in a crowded station due to the people going on and off the train.

Now, let us formally describe the scenario using wireless sensors and detected events, using the notation "sensor description (sensor ID) :: event description (event ID)":

INTELLIGENT CAMERA (S1) ::
EXTENDED PRESENCE ON THE PLATFORM (E1)
INTELLIGENT CAMERA (S1) ::
TRAIN PASSING (E2)
INTELLIGENT CAMERA (S1) ::
PLATFORM LINE CROSSING (E3)
ACTIVE INFRARED BARRIERS (S2) ::
TUNNEL INTRUSION (E4)
EXPLOSIVE SNIFFER (S5) ::
EXPLOSIVE DETECTION (E5)

For the sake of brevity, further steps are omitted.

The composite event **drop of explosive in tunnel** can be specified in EDL as follows:

$$(E_1 \text{ AND } E_2) \text{ OR } E_3 \text{ SEQ } (E_4 \text{ AND } E_5)$$

A partial alarm can be associated to the scenario evolution after step 1 (left AND in the EDL expression), in order to warn the operator of a suspect abnormal behavior. In the design phase, the scenario is represented using Event Trees and stored in the Scenario Repository of DETECT. In the operational phase, SeNsIM records the sequence of detected events in the Event History. When the events corresponding to the scenario occur, DETECT provides the scenario identifier and the alarm level (with a likelihood index in case of non deterministic detection models). Pre-configured countermeasures can then be activated by the SMS on the base of such information.

6.2 Attack of a high-speed railway line

Let us suppose a terrorist decides to attack a high-speed railway line, which is completely supervised by a computer-based control system. A possible scenario consisting in multiple train halting and railway bridge bombing is reported in the following:

1. Artificial occupation (e.g. by using a wire) of the track circuits immediately after the location in which the trains needs to be stopped (let us suppose a high bridge), in both directions.
2. Interruption of the railway power line, in order to prevent the trains from restarting using a staff responsible operating mode.
3. Bombing of the bridge shafts by remotely activating the already positioned explosive charges.

Variants of this scenarios exist: for instance, trains can be (less precisely) stopped by activating jammers to disturb the wireless communication channel used for radio signaling, or starting the attack from point (2) (but this would be even less precise). The described scenario could be early identified by detecting the abnormal events reported in point (1) and activating proper countermeasures. By using proper on-track sensors it is possible to monitor the abnormal occupation of track circuits and a possible countermeasure consists in immediately sending an unconditional emergency stop message to the train. This would prevent the terrorist from stopping the train at the desired location and therefore halt the evolution of the attack scenario. Even though the detection of events in points (2) and (3) would happen too late to prevent the disaster, it could be useful to achieve a greater situational awareness about what is happening in order to rationalize the intervention of first responders.

The formal description of the scenario is the following:

FENCE VIBRATION DETECTOR (S1) ::
POSSIBLE ON TRACK INTRUSION (E1)
TRACK CIRCUIT X (S2) ::
OCCUPATION (E2)
LINESIDE TRAIN DETECTOR (S3) ::
NO TRAIN DETECTED (E3)
TRACK CIRCUIT Y (S4) ::
OCCUPATION (E4)
LINESIDE TRAIN DETECTOR (S5) ::
NO TRAIN DETECTED (E5)
VOLTMETER (S6) ::
NO POWER (E6)
ON-SHAFT ACCELEROMETER (S7) ::
STRUCTURAL MOVEMENT (E7)

Due to the integration middleware made available by SeNsIM, these events are not required to be detected on the same physical WSN, but they just need to share the same sensor group identifier at the DETECT level. Event (a) is not mandatory, as the detection probability is not 100%. Please not that each of the listed events taken singularly would not imply a security anomaly or be a reliable indicator of it.

The EDL description of the above scenario is provided in the following (in the assumption of unique event identifiers):

(((E1 SEQ ((E2 AND E3) OR (E4 AND E5)))
OR
((E2 AND E3) AND (E4 AND E5)))
SEQ E6) SEQ E7

Top-down and left to right, using 4 levels of alarm severity:

a) E1 can be associated a level 1 warning (alert to the security officer);
b) The composite events determined by the first group of 4 operators and the second group of 3 operators can be both associated a level 2 warning (triggering the unconditional emergency stop message);
c) The composite event terminating with E6 can be associated a level 3 warning (switch on back-up power supply, whenever available)
d) The composite event terminating with E7 (complete scenario) can be associated a level 4 warning (emergency call to first responders).

7. Conclusions and future works

New smart-sensor technologies are being investigated by the research community for infrastructure monitoring. These technologies are highly different from each other, and this can significantly contribute to raise detection reliability while lowering the false alarm rate by diverse data correlation [4]; however, this also raises a problem in data integration and reasoning. In this paper we have provided the description of a framework which can be employed to collect and analyze data measured by such heterogeneous sources in order to enhance the protection of critical infrastructures.

One of the research threads we are addressing points at seamlessly integrating sensors and application specific devices which can serve as useful information sources for a superior situational awareness in security critical applications. Furthermore, we are currently extending the EDL language with stochastic operators to allow for heuristic detection model; to this aim, we are also implementing Bayesian Network detection models to account for data and parameter uncertainties.

The next operational step will be the interfacing of the overall system with a real security management system for the on-the-field experimentation. The integration will be performed using web-services and/or the OPC (OLE for Process Communication) standard protocol.

The verification of the overall system is also a delicate issue which can be addressed using the methodology described in [11].

References

[1] Aberer, K.; Hauswirth, M.; Salehi, A. 2006. The Global Sensor Networks middleware for efficient and flexible deployment and interconnection of sensor networks. Technical Report.

[2] Ahn, S.; Chong, K. 2006. Building a Bridge for Heterogeneous Sensor Networks. Proceedings of the Fourth IEEE Workshop on SEUS-WCCIA 06.

[3] Alferes Gaston, J. J. & Tagni, E. 2006. Implementation of a Complex Event Engine for the Web. In Proceedings of IEEE Services Computing Workshops (SCW 2006). September 18-22. Chicago, Illinois, USA.

[4] Bocchetti, G., Flammini, F., Pappalardo, A., Pragliola, C.: Dependable integrated surveillance systems for the physical security of metro railways. In: Proc. 3rd ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC'09), 30 August - 2 September 2009, Como, Italy. To appear.

[5] Casola, V.; Gaglione, A. & Mazzeo A. 2009. A Reference Architecture for Sensor Networks Integration and Management. To appear in Proc. of the 3rd International Conference on Geosensor Networks (GSN 2009), 2009.

[6] Cassandra, A.R.; Baker, D. & Rashid, M. 1999. CEDMOS: Complex Event Detection and Monitoring System. MCC Tecnical Report CEDMOS-002-99, MCC, Austin, TX.

[7] Chakravarthy, S. & Mishra, D. 1994. Snoop: An expressive event specification language for active databases. Data Knowl. Eng., Vol. 14, No. 1, pp. 1–26.

[8] Chakravarthy, S.; Krishnaprasad, V.; Anwar, E. & Kim, S. 1994. Composite Events for Active Databases: Semantics, Contexts and Detection. In Proceedings of the 20th international Conference on Very Large Data Bases (September 12 - 15, 1994).

[9] Dayal, U.; Blaustein, B.T.; Buchmann, A.P.; Chakravarthy, S.; Hsu, M.; Ledin, R.; McCarthy, D.R.; Rosenthal, A.; Sarin, S.K.; Carey, M.J.; Livny, M.; Jauhari, R. 1988. The HiPAC Project: Combining Active Databases and Timing Constraints. SIGMOD Record, Vol. 17, No. 1, pp. 51-70.

[10] Flammini, F.; Gaglione, A.; Mazzocca, N.; Pragliola, C.2008. DETECT: a novel framework for the detection of attacks to critical infrastructures. In: Safety, Reliability and Risk Analysis: Theory, Methods and Applications - Martorell et al. (eds), Taylor & Francis, Proc. of ESREL'08, Valencia, Spain, 22-25 September 2008: pp. 105-112

[11] Flammini, F; Mazzocca, N.; Orazzo, A. Automatic instantiation of abstract tests to specific configurations for large critical control systems. In: Journal of Software Testing, Verification & Reliability (STVR), Wiley, Vol. 19, Issue 2, pp. 91-110

[12] Garcia, M.L. 2001. The Design and Evaluation of Physical Protection Systems. Butterworth-Heinemann, USA.

[13] Gatziu, S. & Dittrich, K.R. 1994. Detecting Composite Events in Active Databases Using Petri Nets. In Proceedings of the 4th International Workshop on Research Issues in data Engineering: Active Database Systems, pp. 2-9.

[14] Gerani, N.H.; Jagadish, H.V. & Shmueli, O. 1992. COMPOSE A System For Composite Event Specification and Detection. Technical report, AT&T Bell Laboratories, Murray Hill, NJ.

[15] Gibbons, P. B.; Karp, B.; Ke, Y.; Nath, S.; Seshan, S. 2003. IrisNet: An Architecture for a World-Wide Sensor Web. IEEE Pervasive Computing, 2(4).

[16] Jones, A.K. & Sielken, R.S. 2000. Computer System Intrusion Detection: A Survey. Technical Report, Computer Science Dept., University of Virginia.

[17] Krishnaprasad, V. 1994. Event Detection for Supporting Active Capability in an OODBMS: Semantics, Architecture and Implementation. Master's Thesis. University of Florida.

[18] Lewis, T.G. 2006. Critical Infrastructure Protection in Homeland Security: Defending a Networked Nation. John Wiley, New York.

[19] M. Rahimi, M.; R. Baer, R. & et al. 2005. Cyclops: In situ image sensing and interpretation in wireless sensor networks. In Proc. 3rd ACM Conference on Embedded Networked Sensor Systems (SenSys'05), 2005.

[20] Perrig, A.; Stankovic, J. & Wagner, D. 2004. Security in Wireless Sensor Networks. In Communications of the ACM, Vol. 47, No. 6, pp. 53-57

[21] Reddy, S.; Schmid, T.; Yau, N.; Chen, G.; Estrin, D.; Hansen, M.; Srivastava, M. B. 2006. ESP Framework: A

middleware architecture for heterogeneous sensing systems. December 2006.
[22] Remagnino, P.; Velastinm S. A.; Foresti G. L. & Trivedi M. 2007. Novel concepts and challenges for the next generation of video surveillance systems. In Machine Vision and Applications (Springer), Vol. 18, Issue 3-4, pp. 135-137.
[23] Roman, R.; Alcaraz, C. & Lopez, J. 2007. The role of Wireless Sensor Networks in the area of Critical Information Infrastructure Protection. Inf. Secur. Tech. Rep., Vol. 12, No. 1 (Jan. 2007), pp. 24-31.
[24] Shneidman, J.; Pietzuch, P.; Ledlie, J.; Roussopoulos, M.; Seltzer, M.; Welsh, M. 2004. Hourglass: An Infrastructure for Connecting Sensor Networks and Applications. Technical Report TR-21-04, Harvard University, EECS.
[25] W3C Architecture Domain, Extensible Markup Language (XML). URL: http://www.w3.org/XML/.
[26] Wiederhold, G. 1992. Mediators in the architecture of future information systems. In IEEE Computer, 25(3).

Author Biographies

Francesco Flammini got with honors his laurea (July 2003) and doctorate (December 2006) degrees in Computer Engineering from the University Federico II of Naples. From October 2003 to January 2007, he has worked in Ansaldo STS as a Software Engineer in the RAMS unit on the verification and validation of real-time control systems. Since February 2007, he has worked in the Innovation unit on the protection of transportation infrastructures. He has been an Adjunct Professor of Software Engineering and Computer Science and currently serves as the Editor in Chief for the International Journal of Critical Computer-Based Systems.

Andrea Gaglione got a B.S. degree and an M.S. degree in Computer Science and Engineering, both summa cum laude, from the Second University of Naples, in 2004 and 2006 respectively. He is currently pursuing a Ph.D. in Computer and Control Engineering at the University of Naples Federico II, since he was awarded a Ph.D. fellowship from AnsaldoSTS in 2006. His research activities are both theoretical and experimental and include sensor networks integration, sensor security and critical infrastructure protection.

Nicola Mazzocca is a full professor of High-Performance and Reliable Computing at the Computer and System Engineering Department of the University of Naples Federico II, Italy. He owns an MSc Degree in Electronic Engineering and a Ph.D. in Computer Engineering, both from the University of Naples Federico II. His research activities include methodologies and tools for design/analysis of distributed systems, secure and real-time systems and dedicated parallel architectures.

Vincenzo Moscato is an assistant professor at the University of Naples Federico II. He received the Laurea degree in Computer Science and Engineering from the University of Naples "Federico II", Naples, Italy, in 2002. His research interests are in the area of image processing (active vision) and multimedia database systems (image databases, video databases, and architectures for multimedia data source integration).

Concetta Pragliola got her laurea and doctorate degrees in Electronic Engineering from the University Federico II of Naples in October 1985. From January 1987 to October 1992 she has worked in the Research Department of Ansaldo Transporti on Expert Systems and Simulation programs. From November 1992 to October 2001, she has worked in the Information Technology Department of Ansaldo Trasporti, being involved in PDM systems. From November 2001 to November 2006 she has worked in Elsag as an Account Manager. Since December 2006 she has worked in the Innovation unit of Ansaldo STS specializing on the design of security systems.