Towards the Improvement of Monitoring and Control Agencies through Knowledge–Based Approaches

Matteo Palmonari, Fabio Sartori
Dipartimento di Informatica, Sistemistica e Comunicazione
Università degli Studi di Milano–Bicocca
Via Bicocca degli Arcimboldi, 8, 20126 Milano, Italy
{matteo.palmonari, fabio.sartori}@disco.unimib.it

***Abstract*—This paper illustrates how the functionalities of monitoring/control systems, within an agent–based framework, can be improved by the adoption of Knowledge–Based techniques. Moreover, the context of Pervasive Computing is chosen as a very promising application domain. After a discussion about the main characteristics a monitoring/control system should have, a possible agent–based architecture for designing this kind of tools is proposed. In particular, it is pointed out how the use of knowledge–based methods is fundamental to provide the systems with the capacity to correlate different perceptions of the reality, in order to obtain more robust, reliable and reasonable system. A possible interpretation of this architecture according to agent theory is given. Finally, a real monitoring/control system based on it is briefly described.**

I. Introduction

Pervasive (or Ubiquitous) Computing (PC) has become a very important topic for researchers in Computer Science. Today, the measurement of physical quantities from the real world and the translation into a digital representation is supported by a lot of sophisticated technologies. Old problems, like size, quality and reliability of measurements, manufacturing and cost of devices have been solved, but new ones are arising. Thus, while most of the industrial and academic research has been focused in past years on the development of efficient communication protocols, now the trend is the design of applications suitable for dealing with the huge amount of information continuously transmitted by devices daily used by a large amount of people.

Basically, PC is concerned with a new way of conceiving the interaction among humans (users) and computing devices. According to M. Satyanarayanan [9], we can look at PC nature as *its ability to disappear into the user subconscious*: mobile devices, sensors and integrated environments depict scenery in which users will interact with embedded devices, each one dynamically connected with each other and almost disappearing.

This extremely general characterization makes PC a *paradigm* rather than a specific research area. To make some clearness, on Lyon's footprints [6] PC can be classified into three wide application areas as long as *personal*, *spatial location* or *biometric* data are collected. The first area deals with Personal Digital Assistants, massively networked devices, smart cards, intelligent textiles, and so on. The second one is about infrastructures concerned with space and spatial location, monitoring, intelligent buildings, and so on. The third is close to the first one, but is related to storing and processing medical and biological information (and this is supposed to introduce new complex, ethical problems).

Within the second research area, we focus our attention on monitoring and control systems. From a general point of view, those systems are expected to collect data by means of sensors and either transmit them to operators or directly use them to perform control tasks, or accomplish both functionalities, supporting humans in their daily tasks. Systems that perform also control activities are much more interesting from our point of view and will be addressed in this paper.

Next section introduces some requirements that a PC monitoring/control system should satisfy in response to some typical problems of pervasive computing. Then, a possible sketch of a suitable architecture for meeting these requirements is given. Section III analyzes this architecture in context of Agent Technology, giving more details about its design. Finally, after a description of a real system developed according to this architecture, some conclusions will be briefly pointed out.

II. Motivations and Background

Pervasive computing introduces some ethical concerns, and this is especially true when notions like "surveillance", a typical issue of monitoring/control systems, seem to be invoked. Some grounded debates on this subject were well developed by A. Stone [11], M. Satyanarayanan [9] and D. Lyon [6][5]. To go into this topic is out of the scope of this paper though.

Nevertheless, paying attention to this sort of debates, we argue that augmenting the global rationality of these systems, e.g. through the refinement of control activity, and focusing on the whole service design it is at least possible to proceed towards a more reasonable scenery for this application area.

Our aim is to provide an architecture that could introduce some elements of the reasonability mentioned above. To achieve this goal, we chose to identify the following minimal requirements a suitable architecture for monitoring/control systems should have:

- *clearness* of the underlying *policy of control*;

- *soundness* according to this policy;
- *flexibility*, in order to model the architecture according to specific contexts and to eventually adapt it to changes in the policy or to new policies;
- *modularity*, in order to satisfy flexibility requirements.

In an attempt to satisfy all the above requirements, we propose to design a multi–layered architecture consisting of four agencies:

1) environmental data acquisition;
2) interpretation of acquired data;
3) correlation;
4) actuation.

We define an agency as an autonomous entity with own operative capabilities and goals, connected with other agencies and viewed in functional terms. Global monitoring and control functions emerges from the interaction of the different agencies.

Data acquisition from the environment is almost performed by *sensors* and related processing. Interpretation of data extracts information according to the specific domain (a sense is given to data depending on the system goal). The correlation agency has the role of managing, filtering and correlating information coming from the interpretation agency, so that:

- a *global view* of the monitored situation that contextualizes different *local* interpretations is created;
- control actions are taken according to this global view, through the adoption of a precise and explicit control policy;

It is important to highlight that the control policy of the system should be modelled by the core knowledge concerning the domain. Thus, we hold that knowledge–based approaches are particularly suitable for the design of correlation agency. Finally, actuation agency is committed to effectively carry out the control task, influencing the environment.

Currently, many monitoring and control technologies take control actions only on the basis of local interpretations of data collected by sensors. Thus, they tend to be brittle and often ineffective, since they are too subject to noise, possibly triggering not necessary or even undesirable control actions (e.g errors of detection or inference). What they miss is an agency correlating local information (i.e. local interpretations of data collected by sensors) at a higher level. A multi–layer architecture like that one sketched above improves the functionality of the system providing the contextualization of local interpreted reports, allowing a global monitoring and a more reasoned control.

III. A Four Level Agent–Based Architecture for Monitoring and Control Systems

In the section before, we introduced the notion of agency in the context of pervasive computing, focusing on monitoring and control systems. This means that we are interested in the development of agent–based frameworks: although talking about agencies seems to be quite natural in order to respond to high level constraints like those shown above, this choice requires anymore clarifications. In particular, we should at least explain:

- why we introduced an agent–based framework and why it is useful in our context;

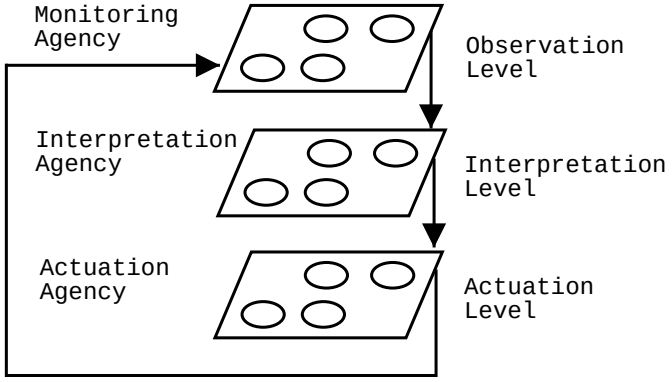

Fig. 1. The general architecture of a monitoring/control pervasive system

- what we precisely mean by the term “agency”, intuitively and synthetically defined above.

A simple answer to the first question is that monitoring and control systems usually works in (often widely) distributed environments and this makes the agent–based approach extremely useful [4]. Moreover, distributed elements of the systems have generally different tasks and need to cooperate to achieve a global purpose. From this point of view, an agent–based approach is not absolutely required, but it offers a convenient and handy framework to work within.

In control and monitoring systems, there are usually different devices (some of which embedded in the environment) that are clearly autonomous, have separated concerns and need to communicate in order to cooperate towards the global goal of the system. The communication need not to be necessarily performed by means of a high level language, because the interpretation process often passes through different levels of abstraction, but it needs to be strongly structured according to the different functions of the agencies. For example, the interpretation of pixels returned by video–cameras (i.e. sensors) is autonomous and has not to be interested in the further processing of the information provided, but this last one must be transmitted to the higher level not as simple data but as already interpreted data.

Properly, the objects we call agencies aren’t necessarily agents, according to a strong definition of that term [13], but we intend to use the wider concept of agency when the context appeals to a high level of abstraction. “Agency” is a functional concept: it is thought to meet a set of specifications and to perform some functions that, taken together, respond to a given definition of agent (e.g that found in [4] [13] or [3]). We mean to distinguish unitary entities specifically designed as agents from conceptual entities whose behavior can be *assimilated* to that of an agent. Thus, an agency can be variously implemented until its behavior responds to the given definition; obviously, one or more agents represent a natural way by which an agency can be developed. More precisely, in the last case, we can identify two possible choices: an agency can be implemented by agents

1) that have the same functionalities as the agency itself;
2) whose interactions and combined working allow the emerging of the agency functionalities.

For example, given a system S and a data–collector agency A, with features [*x*,*y*,*z*], this can be implemented (1) by many collector agents whose single features are the same [*x*,*y*,*z*] of A (in this case the agency's specifications almost collapse on the ones of the single agents) or (2) by agents with different functionalities whose interaction accomplishes the data collecting task of [*x*,*y*,*z*] (we can have, for example, three agents whose features are respectively [*x*], [*y*], [*z*], but we can also have more complex combinations).

On the other hand, we may have the case in which the same data collecting task (almost indistinguishable from a higher level of abstraction) is carried out by a software architecture whose elements are not agents themselves (e.g. they do not have any social ability - but simple interface protocols - or they are not proactive) but whose global behavior is *comparable* to that of an agent. With reference to the previous example, if the combined work of a set of programs covers the features [*x*,*y*,*z*] mentioned in the last paragraph, we can refer to that *set* of programs as the agency A.

This conceptual distinction between agent and agency is interesting because it also provides two different meanings of *agent–based model*: strictly speaking, an agent–based model refers to an architecture whose main components are agents. But, from a higher point of view, we can also talk about agent–based models when the architecture can be *functionally* viewed as made up of different agencies. Therefore, these two notions of architecture are referrable to two different levels of abstraction: the agent–based models in the first meaning are also agent–based models in the second meaning, but the converse does not hold.

What previously said is subsumable in the following way: given a definition D for "agent", an agency is whatever behaves like the agent defined by D (of course, different definitions of "agent" put different constraints over the respective notions of agency). Speaking of an agent–based model from a functional point of view appeals to the notion of agency, while the strong meaning of the term refers to how the agencies are built and to what they are made of.

Currently, our position is to propose an agent–based architecture (at least) in the functional meaning. This provides an interpretative lens to look at distributed and modular control and monitoring systems within a well defined framework; moreover, reasoning on single autonomous agencies makes the characterization of system specifications easier and the control of their fulfillment clearer (especially as long as high level specifications are concerned, like those introduced in section II).

Today, most monitoring and control systems are structured on three logical levels, shown in Figure 1:

- *observation*, where the state of a monitored field is periodically captured by a specific *monitoring agency* (MA), usually a set of sensors;
- *interpretation*, where values detected by sensors are evaluated by a specific *interpretation agency* (IA);
- *actuation*, where specific actions are taken by a specific *actuation agency* (AA), depending on the interpretation results.

The most critical step in the process above is the interpretation of data: an error could cause the system takes a wrong decision. Although a lot of sophisticated technologies have been developed, and efficient detection algorithms have been implemented, interpretation of complex data, like images, smells and so on is still a problem. Moreover, monitoring and control systems have only a local perception of the environment, while a global one would be more suitable to avoid errors or imprecisions in the interpretation.

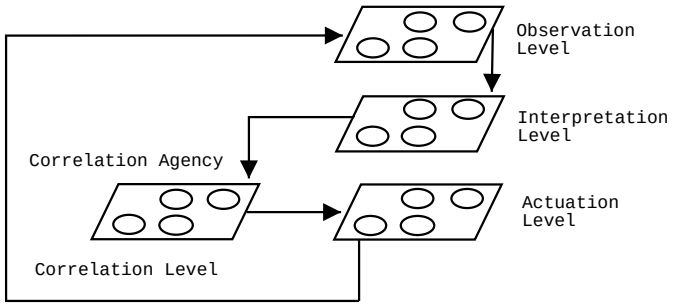

Fig. 2. The general architecture of a monitoring/control pervasive system, extended with the correlation level

A possible solution of this problem is shown in Figure 2, where a fourth logical level has been added between the interpretation and actuation ones. This level is characterized by *correlation*: interpretations of data are correlated, by a specific *correlation agency* (CA), in order to identify possible mistakes and avoid the generation of wrong decisions. The introduction of correlation level in a pervasive system architecture causes a change in the functionalities of interpretation level, switching from *take a decision* to *suggest a decision*. All the suggestions are evaluated by correlation level, so that a monitoring/control system will chose the proper actions to be adopted according to a *global* view of the state detected by sensors rather than a *local* one.

The correlation is guided by a specific *policy*, depending on the kind of problems to be solved. The policy should at least specify:

- actions to be taken in response to a global critical situation;
- rules to correlate different local situations into a global one.

CAs can be designed and implemented exploiting a lot of technologies, such as *Knowledge Based Systems* [12] (KBS), *neural networks* [10], *data mining* [1], and so on. Probably, the most suitable paradigm for the realization of the architecture above is the *agent–based model*. In this case, control and monitoring agency can be considered as set of agents, with at least one element.

In particular:

- elements of MA, IA and AA are typically represented by *reactive agents*. Each of them receives an input and makes an action immediately: for MAs, the input is the observation of local state and the action is its transmission to IAs, for IAs the input is the value received from MAs and the action is to suggest a decision, for AAs the input is the decision taken by CAs and the action is to do it;
- CA members are typically represented by *utility–driven agents* [8]. In fact, they receives all the evaluations of local state from interpretation agency, correlates them and make

Type	IA Card	CA Card	Problem Complexity
one–to–zero	1	/	low
n–to–one	n	1	Medium
n–to–n	n	n	High

TABLE I
A PERVASIVE SYSTEM CATEGORIZATION

deliberations about the proper actions to be taken. Then, they notify actuation agencies about their decisions.

The reasoning of correlation agencies depends on the type of policy, that is a sort of *utility function*. For instance, a very suitable representation for a policy is a rule–based system, in which the rules are activated by the local state values.

Interpretation and correlation agencies can result more or less populated in a pervasive system. In particular:

- an IA must be always present, while the existence of a CA is not mandatory;
- if the IA is made of only one agent, then CA is not necessary, since there are no correlations to be done;
- if an IA has two or more elements, then the presence of CA is recommended;
- the cardinality and structure of CAs depends on the problem complexity.

Table I summarizes the categorization of pervasive systems above according to three different levels (low, medium and high) of problem complexity.

The four–level architecture introduced above is general enough to be applied to a lot of real systems, developed in the context of critical domains. In the next section a pervasive computing system based on it is illustrated. In particular, the system presented belongs to the *n-to-one* category (that is a very common scenario).

IV. SAMOT, A SYSTEM FOR AUTOMATIC MONITORING OF TRAFFIC

One of the most interesting application of pervasive computing is traffic monitoring and control. The aim of a traffic monitoring and control system is to identify traffic flow anomalies, to alert operators and to support them in the management of emergencies [2].

Such systems can exploit a lot of sophisticated technologies for traffic anomalies detection, like magnetic–loop sensors, infrared, video image processors and so on. In particular, the adoption of video–based technologies has allowed to increase the effectiveness of systems in traffic monitoring and control, providing them with the possibility to clearly distinguish different critical situations. Unfortunately, there are many situations in which video–based technology can be misled, generating false alarms about traffic conditions.

This problem has been solved in the SAMOT Project through the development of a knowledge–based Module for the Correlation of Alarms (MCA) coming from intelligent video–cameras. SAMOT is a monitoring/control system developed in collaboration with Project Automation S.p.A. and Società Autostrade (i.e. the Italian Highway Company), currently installed and working on some Italian highways.

Figure 3 shows the four–level architecture of the system.

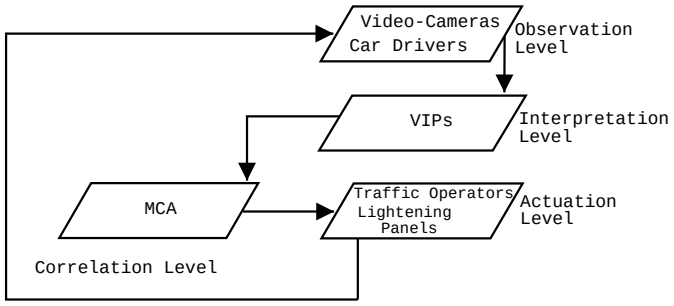

Fig. 3. The four–level architecture of SAMOT

In SAMOT, the *monitoring agency* is made of a set of video–cameras, each one devoted to the surveillance of a small portion of the monitored road. A Video–Image Processor [7] (VIP) is associated to every camera, in order to make an interpretation of the traffic situation. The set of VIPs is the *interpretation agency* of SAMOT. The MCA is the *correlation agency* of the system, and it will be further described in the next section. Finally, traffic operators and lightening panels placed along the road are the elements of *actuation agency*: according to the output of MCA, traffic operators (or the MCA itself) activate one or more lightening panels, showing warning messages to car drivers.

V. THE ALARM CORRELATION MODULE

MCA supports traffic operators in the interpretation of traffic situations, correlating atomic traffic anomalies and filtering them taking into account their *spatial* and *temporal* location. MCA is a *knowledge–based module* acting as a goal–driven agent: its utility function implements a policy, that is a set of rules expressing the expertise of operators.

The main entities involved in the correlation process are the *monitored highway*, *alarms* coming from VIPs and *Anomalous Traffic Patterns* (ATP), complex events dynamically built up starting from alarms. In the SAMOT model, the highway is viewed as a composition of spatially adjacent and regular sectors, each one monitored by a single video–camera with a related VIP generating alarms. The management of ATPs represents how the correlation agency of SAMOT works, and it is the result of a deep knowledge acquisition campaign.

The last function of MCA is to filter alarms coming from VIPs. On the basis of spatial and temporal correlations, MCA is able to create a global view of the traffic situations, recognizing possible false alarms and supporting traffic operators in identification of real problems. Then, traffic operators or the MCA itself can undertake the proper actions to solve them (e.g. activating the correct lightening panels and notifying car drivers).

Figure 4 shows a high–level view of how MCA works with respect to the other agencies SAMOT is composed by.

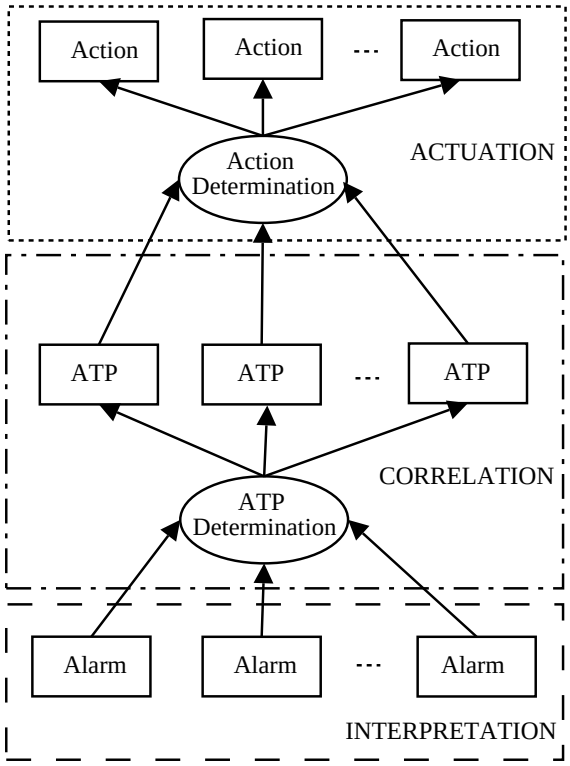

Fig. 4. Realtionships between MCA and other SAMOT agencies.

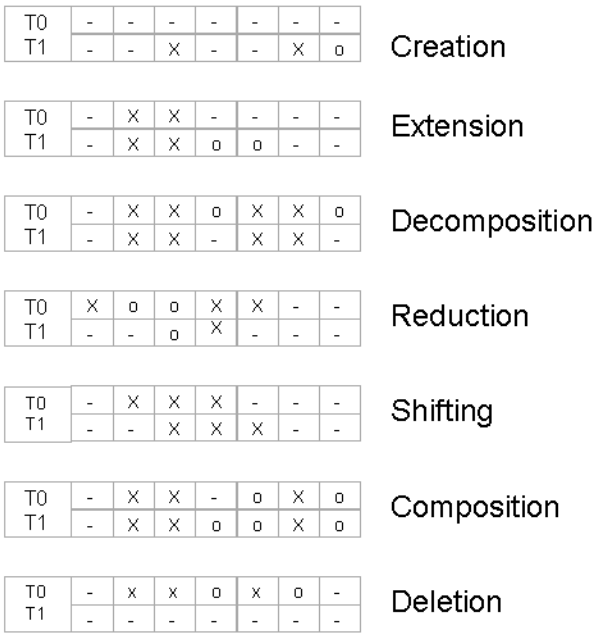

Fig. 5. A representation of ATPs.

A. From Interpretation to Correlation: Anomalous Traffic Patterns

An anomalous traffic pattern represents a traffic anomaly referring to multiple cameras and, consequently, a sequence of local traffic situations detected by VIPs.

The creation of ATPs reflects precise rules of *spatial adjacency*: two or more local spatially adjacent anomalies can be merged by MCA. Then, ATPs are compared by MCA on the basis of *temporal adjacency*: for instance, if two or more temporally near ATPs reports a common anomaly, such as slow traffic, MCA can deduce that there is a queue caused by a car accident.

The development of MCA has been fundamental: in fact, despite of their high level technology, VIPs often fail the evaluation of traffic conditions, due to a lot of reasons, generating false alarms about critical situations. The main reason for this is that VIPs don't work exploiting specific knowledge: their interpretation is limited to the analysis of images supplied by video–cameras, by which they're able to recognize possible anomalies, but their response can be negatively influenced by a lot of natural and unpredictable factors (i.e. shadows on the road–ground, water and so on).

An ATP can be considered as a set of adjacent highway sectors characterized by common local conditions (i.e. normal traffic, queue, incident, slow traffic and so on). Moreover, the distribution of ATPs over the road dynamically changes: an ATP characterized by *queue* at certain instant could modify its state to *normal traffic* in the future. Figure 5 shows the basic operations for the management of ATPs in SAMOT.

Each column concerns a portion of the road section monitored by cameras referring to the same VIP. Circles and crosses respectively indicate 'slow traffic' and 'queue' atomic traffic anomalies, while empty cells represent 'normal' traffic situation. Each pair of rows of Figure 5 schematically represents two traffic patterns that is, two sequences of atomic traffic anomalies, referring to two consecutive time stamps (i.e. T0 and T1). As shown in the figure, according to spatial and temporal relations among atomic traffic anomalies and anomalous traffic patterns, anomalous traffic patterns can be:

- *Created*, when two or more spatially adjacent road section portions change their states from 'normal traffic' (i.e. empty cells at T0) to 'anomalous traffic' (i.e. circles and crosses in T1);
- *Extended*, when at least a road section portion that is adjacent to an already detected anomalous traffic pattern changes its state from 'normal' to 'anomalous' traffic;
- *Decomposed* into two patterns, when at least a road section portion belonging to an anomalous traffic pattern, changes its state from 'anomalous' to 'normal' traffic;
- *Reduced*, when at least a road section portion that is one of the endpoints of an anomalous traffic pattern changes its state from 'anomalous' to 'normal' traffic;
- *Shifted*, when an anomalous traffic patterns is reduced at an endpoint and extended at the other endpoint; item *Composed* with another pattern, when at least a road section portion that is located between two different anomalous traffic patterns changes its state from 'normal' to 'anomalous' traffic;
- *Deleted*, when all the road section portions of an anomalous traffic pattern change their states from 'anomalous' to 'normal' traffic.

B. From Correlation to Actuation

```
if
   (Initial state = RegularTraffic(k))
   % portion k is characterized by 'Regular Traffic'
   (Anomaly = VStopped(k), ts)
   %'Stopped Vehicle' detected on k at time ts
then
   (ActionsGUI = VStopped(k))
   % 'Stopped Vehicle' anomaly shown on operator GUI
   (Beep = 'yes')
   % acoustic warning to operators
   (Video = 'k')
   % images on CCTV are fixed on the portion k
   (Panels(k) = 'Incident')
   % adequate message for VMPs are created
   (Time = ts)
   % immediate operator alerting
   (Duration = 'Event')
   % control actions end when the anomaly ends
```

The correlation of atomic traffic anomalies allows an interpretation of traffic situation similar to the one usually performed by traffic operators. The MCA module of SAMOT also interacts with the actuation agency of the system in taking decisions to solve critical decisions. To do this, supports human operators by generating warnings that will be shown on lightening panels, according to an opportune interpretation of ATPs.

As an example of anomalous pattern interpretation, let us consider the detection of 'stopped vehicle' after a regular traffic situation and the same anomaly detected when a traffic congestion is already present. A dedicated set of MCA rules represents the knowledge for traffic anomaly interpretation taking into account spatial and temporal dependence among different traffic patterns. For instance, in rule reported above, temporal dependence is taken into account in the interpretation of the traffic anomaly 'Stopped Vehicle'.

Different interpretations are given to this traffic anomaly depending on the traffic situation referring to the time stamp immediately preceding its detection. For instance, if the anomaly is detected after a 'Regular Traffic' situation, operators are alerted that an incident may have caused the 'Stopped Vehicle' traffic anomaly.

Thus, the final output of MCA is a sequence of possible actions (i.e. visual or acoustic warning to operators, localization of highway sectors involved in the critical situation, visualization of warning message on lightening panel to notify drivers) to be taken in order to solve complex anomalies represented by ATPs. An action can be:

- *Created*, when a new ATP has recognized;
- *Deleted*, when the ATP it was created to deal with has finished to exist;
- *Modified*, if an existing ATP should be managed in a different way.

Table II shows some possible actions depending on different anomalous situations: S(K) indicates that a vehicle is stopped on the sector monitored by the k–th video–camera, WW(k) that a vehicle is proceeding against the right direction of traffic flow, and C(k) that the sector is characterized by slow traffic or queue. Rows from 4 to 10 concern generation alarm successions, in which the first one existed on the k–th highway sector before the second one. The 8–th case is particularly interesting, since it takes care of a stopped vehicle at the end of a congestions: the MCA module deduces that a car accident happened on the k–th road sector, preferring to signal an *Incident warning* with respect to a *Queue warning*.

ATP	**Visual Warn**	**Acoustic Warn**	**Monitor**	**Light. Panel**
S(k)	Stopped Vehicle	Yes	k	Incident
WW(k)	Wrong Way Vehicle	Yes	k	Incident
C(k)	Congestion	Yes	k	Queue
S(k), WW(k)	Wrong Way Vehicle	Yes	k	Incident
WW(k), S(k)	Wrong Way Vehicle	No	k	Incident
S(k), C(k)	Congestion	No	k	Queue
C(k), S(k)	Congestion	No	k	Queue
C(i,j,k), S(k)	Stopped Vehicle	Yes	i,j,k	Incident
WW(k), C(k)	Congestion	No	k	Queue
C(k), WW(k)	Wrong Way Vehicle	Yes	k	Incident

TABLE II
ACTIONS TAKEN BY MCA DEPENDING ON ATPS

VI. CONCLUSIONS

The paper has presented a four–level architecture for the development of monitoring/control systems.

We established four basic requirements that such systems should take into account in order to provide a more reasonable service. The architecture proposed allows to localize the core knowledge of the domain (in the correlation agency), thus making the control policy explicit and clearly specified. Firstly, this feature enables to fulfill the requirements of *clearness* and *flexibility*. In fact, the correlation agency requires the explicitness of a control policy. Moreover, it identifies the layer (i.e. the correlation level) where substantial modifications should be made to adapt the architecture to different policies or to eventual changes. The agency model is typically *modular* and an agent–based approach empower this characteristic. Finally, *soundness* with respect to the control policy model depends, obviously, on different implementations, but the architecture put at least the basis for an easier and clearer correctness checking.

The correlation agency is the element that improves robustness, reliability and reasonability of a monitoring/control system, since it exploits the core knowledge of domain experts. As an example, the MCA module of SAMOT has been introduced. The addition of MCA to the architecture of the system has allowed to avoid interpretation errors by VIPs, that were quite frequent.

The problem of privacy in monitoring/control system is a fairly controversial issue related to the concerns about "surveil-

lance" mentioned so far. Understanding if adequate arrangements in the interpretation and, especially, correlation agencies design can partially respond to this problem could be subject of future works.

REFERENCES

[1] S. Brossette, A. Sprague, W. Jones, and S. Moser. A data mining system for infection control surveillance. Technical report, Department of Pathology, University of Alabama at Birmingham, USA, 2000.

[2] N. J. Ferrier, S. M. Rowe, and A. Blake. Real–time traffic monitoring. In *WACV94*, pages 81–88, 1994.

[3] M.R. Genesereth. Software agents. *Communication of the ACM*, 37(7):48–53, 1994.

[4] N.R. Jennings. On agent-based software egineering. 117:277–296, 2000.

[5] D. Lyon. *The Electronic Eye: The Rise of Surveillance Society*. Polity-Blackwell, 1994.

[6] D. Lyon. *Surveillance Society: Monitoring Everyday Life*. Open University Press, 2001.

[7] M. Egmont Petersen, D. de Ridder, and H. Handels. Image processing with neural networks: a review. *Pattern Recognition*, 35(10):2279–2301, 2002.

[8] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Engelwood Cliffs, Nj, 1995.

[9] M. Satyanarayanan. Privacy: the achilles heel of pervasive computing? *IEEE Pervasive Computing–Mobile and Ubiquitous Systems*, 2003. Available at http://www.computer.org/pervasive/pc2003/b1002.pdf.

[10] D. Sobajic. Applications of neural networks in environment, energy and health. Progress in Neural Processing 5, chapter 11. World Scientific Publishing Co. Ptd. Ltd., P. O. Box 128, Farrer Road, Singapore 912805, 1996.

[11] A. Stone. The dark side of pervasive computing. *IEEE Pervasive Computing–Mobile and Ubiquitous Systems*.

[12] R. L. Westra. Design of a knowledge–based monitoring and control system. Technical report, Department PRG/Faculty Math. & Comp. Sc., University of Amsterdam, 1989. Esprit 2439 report D31.

[13] M. Woolridge and N.R. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2):115–152, 1995.