Multimedia Tools and Applications manuscript No.
(will be inserted by the editor)

A Context-aware Middleware for Real-Time Semantic Enrichment of Distributed Multimedia Metadata

Nikolaos Konstantinou · Emmanuel Solidakis · Anastasios Zafeiropoulos · Panagiotis Stathopoulos · Nikolas Mitrou

Received: date / Accepted: date

Abstract This paper investigates the problem of the real-time integration and processing of multimedia metadata collected by a distributed sensor network. The discussed practical problem is the efficiency of the technologies used in creating a Knowledge Base in real-time. Specifically, an approach is proposed for the real-time, rule-based semantic enrichment of lower level context features with higher-level semantics. The distinguishing characteristic is the provision of an intelligent middleware-based architecture on which low level components such as sensors, feature extraction algorithms, data sources, and high level components such as application-specific ontologies can be plugged. Throughout the paper, Priamos, a middleware architecture based on Semantic Web technologies is presented, together with a stress-test of the system's operation under two test case scenarios: A smart security surveillance application and a smart meeting room application. Performance measurements are conducted and corresponding results are exposed.

Keywords Semantic · Middleware · Rule-based · Real-time · Context-aware · Annotation.

1 Introduction

Currently existing multimedia data, either publicly available or in restricted access areas, should be annotated in order to become easily and meaningfully retrievable. This purpose is served by a number of prevalent Semantic Web technologies like content description languages, query and rule languages, and annotation frameworks. These technologies provide the common framework for human and machine consumption of data. Specifically, Semantic annotation in the form of metadata can be added to any form of context, in order to add well-defined semantics that will enrich the context information and boost its (re)usability.

National Technical University of Athens
Heroon Polytechneiou str., 15773, Zografou, Athens, Greece
Tel.: +30-210-7722425
E-mail: nkons@cn.ntua.gr

Despite ongoing research efforts on advanced content-based annotation and retrieval techniques in A/V repositories, keyword-based annotation approaches still prevail in content description and thus, content retrieval. Hugely popular online services employ it such as flickr.com for pictures, del.icio.us for user bookmarks, youtube.com for video and last.fm for audio content. In any case, annotation is typically kept separately from the data, in the form of metadata, an approach that is compliant with the Semantic Web model.

Given the importance of Semantic annotation, one could wonder why its presence is not always guaranteed. The answer is a combination of several factors. Firstly, it is a time-consuming task. Users lack time or do not consider it important enough to spend time annotating already published content. The service providers on the other hand mostly believe that annotation is a loss of resources in terms of time and money. Moreover, it is an error-prone task that requires expertise in order to be conducted properly. Also, the reuse of this information is troublesome as annotation is usually likely to be redundant, partial or stored in different formats [1]. If we take also in consideration that annotation easily becomes outdated then we can easily state that without automation, the future of the Semantic Web still has to face many challenges [2].

The automation of the whole annotation procedure is a required step further towards its wider deployment. Especially in context-aware applications which typically involve large volumes of multimedia data and its metadata, the need of real-time automatisation of the annotation procedure is indisputable. User friendly configuration of the annotation system based on different application needs is another important dimension that needs to be addressed.

On the other hand, even annotated content is not directly exploitable by users as the development of inference-based applications for content delivery to different user categories is still a painstaking process. This is more obvious in cases where large bulks of real-time A/V and contextual data are produced from several data sources (e.g. sensors) and need to be processed both real-time and offline. For example, an advanced security surveillance system should be able to cope at a Semantic level with all the following:

- Reliable real-time alerts produced in emergency situations decided during system operation (e.g. an unknown person is entering a restricted area without being accompanied by an identifiable person),
- Dynamic system configuration and adjustment both to the environmental conditions in time and space (e.g. dynamic sensor activation/deactivation, camera selection depending on the point of interest), and to the underlying network conditions and user devices (e.g. selection of the quality level of an information flow based on its Semantic content),
- Acceleration of offline searching of important facts (e.g. identification and movement tracking of a suspect person entering a restricted area in a time zone of high risk, identification of an object left for more time than allowed in a restricted area).

What we present in this paper is Priamos, an open, rule-based middleware system for real-time Semantic enrichment of context features, which enables the development of inference based applications, as it is implemented using Semantic Web technologies. The distinguishing characteristic of the Priamos architecture is that both low level components, such as sensors, feature extraction algorithms and data sources, and high level components, such as application-specific ontologies and rule sets are pluggable

to the middleware architecture, while high-level interfaces are exposed for prototyping and development of innovative applications that require inference. Furthermore, we demonstrate how the logic of an application can be modeled on top of the middleware components in order to launch a context-aware system that will provide real time higher-level Semantics and create a Knowledge Base (KB) where the system's awareness of the world will be nested for offline searching.

The paper is structured as follows: Section 2 presents the motivation behind the current work, the state of the art in the field of content annotation and context-aware inference-based applications and a comparison our approach with the related work. Section 3 analyzes the overall architecture and the software implementation. Section 4 demonstrates two test cases of the system, operating under an advanced security-surveillance application scenario and a smart meeting room scenario in a laboratory environment, while a performance evaluation is presented in Section 5. Finally, Section 6 concludes the paper by noticing the future directions for expanding the presented work.

2 Motivation and Related Work

Firstly, we should explain why the use of MPEG-7 or other similar approaches would not suffice and what does the architecture here offer in addition. The need of a higher-level capturing of the semantics of a multimedia document has led to the establishment of the MPEG-7 standard but, unfortunately, the standard was not designed with the Semantic Web community in mind [3], a fact that is obvious since the description language is in XML. Therefore, for the needs of the hereby presented work, the MPEG-7 standard is not sufficient. The necessity of homogenizing distributed video streams on a higher level led us to adopt a purely semantics-based approach. As shown in Section 3, the approach presented here can describe events captured from multimedia sources in as high level as the scenario demands. The adoption of MPEG-7 would restrain the system's knowledge to the standard's relatively poor semantic boundaries.

Semantic Web-compliant standards have been proposed such as the OWL-based VERL and VEML [4]. These standards are employed in order to annotate and record objects and (sub)events in video streams. The restrictive characteristic of this work in comparison with the hereby presented work is the use with video information. Our approach defines a more generic approach that can handle incoming flows of other types of information; not only video. In Section 3.2, we will discuss about how the proposed approach offers the option of employing standards such as VERL, or Adobe's open, standards-based XMP[1] or any Semantic Web vocabulary or microformat in order to satisfy applications' needs in describing their context.

We continue by providing the definition of *real-time* and *context-aware* notions that are the key concepts of the Priamos approach.

2.1 Real-Time Processing

According to [5], all systems can intuitively be made to look as if they were real-time simply by defining arbitrary deadlines to conform to. But, the behaviour of an actual

[1] Extensible Metadata Platform (XMP): `http://www.adobe.com/products/xmp/`

real-time system is defined in terms of the system's tolerance to missed deadlines. A real-time system is one that must satisfy explicit bounded response time constraints to avoid failure and present consistency regarding the results and the process time needed to produce them. This distinguishing feature of the real-time systems is called *timeliness*. The time that elapses between the presentation of the input and the presentation of the output is called *response time* or *latency*.

The fundamental difference between a real-time and a non real-time system is the emphasis in predicting the response time and the effort in reducing it. Actually, there is a misconception that the real-time systems should respond in fast times (i.e. microseconds). But the deadlines of a real-time system in fact are depending on the underlying processes being controlled.

In general, real-time systems are divided into three kinds. In *hard real-time* systems, failure to meet even one deadline results in total system failure. In *firm real-time* systems, a small number of deadlines can be missed without total system failure. Third, in *soft real-time* systems missed deadlines cause degradation of performance; not failure.

In this paper, the approach presented constitutes a hard real-time system, in the sense that the deadlines are respected and system failure is avoided. As demonstrated in the conducted measurements in Section 5, when the system's response rises above certain thresholds, maintenance procedures take place in order to assure timeliness.

2.2 Context information

As denoted in the title, the presented middleware can offer the basis for context-aware applications. Therefore, it should be clarified what does the existence of context entail: Context means situational information. According to [6], context is "...any information that can be used to characterize the situation of an entity". An entity can be a person, a place or an object that is considered relevant to the interaction between a user and an application. The user and the application are considered as entities as well. A system is context-aware if it can extract, interpret and use context information and adapt its functionality to the current context of use.

Throughout this paper, we will use the words *context information* or *multimedia content* indistinctly because they both refer to the medium through which the system gains awareness of its surrounding world. In the work presented here, the multimedia streams actually are the primitive form of the information that flows towards the middleware comprising its context.

Sensors and appropriate software are required to capture context information. A common representation format should be followed, in order to transfer the contextual information to applications and in order for different applications to be able to use the same information basis. Thus, in order to ensure syntactic and semantic interoperability, one solution would be a Semantic-Web-compliant approach regarding annotation that enables the aggregation of various heterogeneous data sources.

Nevertheless, content annotation does not constitute a single panacea for content retrieval and direct utilization by applications. The reason is that no single approach exists as far as homogenizing bulks of data is concerned with. Standards such as MPEG-7 offer the means to annotate multimedia data, however, it remains as a question how accurate the metadata are, how convenient is it to maintain in accurate state and, mostly important, what the true added value to the content is. In other words it is not clear how can the user benefit from the existence of such metadata. In order to provide

a viable annotation mechanism, there is need for the quickest automated annotation that can offer the highest possible level of intelligence. Ease of development leads to lower costs and faster times in application development and system maintenance.

Let us consider as an example the cameras that monitor public areas such as metro stations that provide live streams of multimedia data, or even multimedia repositories owned by television or radio broadcasters that might be archived. The use of keyword-based technology does not facilitate the creation and execution of intelligent queries such as:

- When was a person or one of his peers last seen?
- When was the last time a person was in a certain area without the attendance of a supervisor?
- When were persons X and Y simultaneously in the same place?

These queries should also be able to trigger alerts; a system's desired feature would be to produce a notification if one of these queries returns any results.

As far as context-aware systems are concerned, world concepts can be described in detail using Semantic Web technologies. Most of the potential power in these approaches is that a world model can be bound to a reasoner and deduce implicit knowledge, adding intelligence to the system. However, context-aware systems are often complicated enough to the point that tasks like annotation and decision making become unmanageable if not supported by automated procedures. Moreover, ad hoc formalisms with insufficiently established semantics make context aggregation difficult [7]. But, the use of a middleware facilitates context representation and processing at the infrastructure level, enabling the better reuse of derived context by multiple data consumers.

By closely investigating the contextual information processing procedure, we notice that there is need of a unification of the feature extraction algorithms under a common terminology, in order to produce powerful results. Data that is collected by the sensors is very often in a custom format. To create a context aware application based on this received data flow, the data itself will pass through many stages until it is converted from electromagnetic signal to human understandable information. We need to define the boundaries of these processing steps along with the interfaces needed for the communication of the various levels. Accordingly, we need to identify the reusable components that can comprise a middleware without having to be reimplemented for each application that uses them. This is exactly the purpose of the hereby presented middleware.

Moreover, following the proposed implementation, problems such as false alarms that can be produced by tracker errors can be dealt with. Even if an event erroneously escapes from the tracker's implementation, it can be dealt with at a higher semantic level. For instance, suppose that a tracker mistakably recognizes an object and sends the information to the middleware. A rule at the semantic level can state that if this object did not have a continuous presence for e.g. two seconds, it can be a probable error of the tracker.

It should be noted, however, that users of automatic annotation systems need to be aware of their limitations. Broadly speaking these are missing annotations (known technically as low recall) and incorrect annotations (known as low precision), and they trade off against each other. However, organizations with large collections of legacy data in particular, may prefer imperfect annotation than no annotation at all [8].

2.3 Related Work

The work presented in this paper on one hand can be considered as an approach to automated multimedia content annotation. On the other hand, the Priamos middleware offers an infrastructure for building context-aware applications. Both points of view concern semantic information enrichment. Thus, in the related work presented in this Section we gather approaches that present conceptual similarities to the Priamos approach.

Regarding the general domain of automated content annotation, we can observe that it is highly active, with many suggested approaches, each with its own benefits and drawbacks. According to [9], the information that a multimedia document conveys can be formalized, represented and analyzed with three levels of abstraction: subsymbolic, symbolic and logical.

The first level targets the raw data represented in well-known formats for video and/or audio. A variety of algorithms exists for automatically recognizing and tracking specific features such as an object or a human face in video streams, or speech in audio streams. Regarding face detection for instance, the Viola-Jones algorithm [10] can be used. Similarly, for face recognition one can employ the PCA [11], LDA [12] or ICA [13] algorithm, to name a few. Regarding face or object tracking, an algorithm such as the Mean-shift [14], the Camshift [15] or a Kalman filter [16] can fulfill an application's needs. Also, speech recognition algorithms can be employed such as [17]. In addition, several methods have been proposed in the bibliography for extracting events of interest in audiovisual streams. In [18], an annotation mechanism for basketball games is presented, while in [19, 20] the targeted sport is baseball.

While the majority of these approaches can provide powerful results, annotation at this level is not necessarily suitable for further processing in the sense that it cannot cover the needs for integration of the essentially heterogeneous information originating from various implementations. The next (symbolic) level addresses this issue. Many standards such as MPEG-7, MPEG-21, Visual Resource Association (VRA), International Press Communications Council (IPTC), NewsML and so on, mainly operate at this level. The purpose is to provide homogeneity in annotations in order to combine the emerging results. However, the problem with these approaches is that each annotation effort is syntactic – in the sense that only a structural approach is provided in each case – and tightly coupled with the characteristics of each vocabulary and the corresponding multimedia stream format. This does not allow integration and interoperability between annotated content. Therefore, the semantic (logical) layer is needed.

The third layer offers semantic enrichment of the annotated information by making use of Semantic Web technologies. Tools in this category include for instance Vannotea [21] that can annotate collections of multimedia files, M-Ontomat Annotizer [22] and the Ontology-based annotation system AKTive Media [23]. Also, user-centered approaches for multimedia annotation have been presented, such as Armadillo [1], KnowItAll [24] and the SmartWeb[2] project where an unsupervised approach for RDF Knowledge Base population is investigated.

As far as the pattern-based and rule-based approaches are concerned, we can see only a few of them in the bibliography. These include CAFETIERE [25], a rule-based system for generating XML annotations that was developed as part of the Parmenides

[2] The SmartWeb project: `http://smartweb.dfki.de/`

project and does not make use of Semantic Web technologies. The OntoLT Protégé plugin [26] offers ontology extraction from text. In order to extract knowledge, it provides mapping rules, defined by use of a precondition language that allow for a mapping between linguistic entities in text and class/slot candidates in Protégé.

The PICSEL project [27] has passed through several stages of development and is still evolving, with PICSEL 3 starting in 2005[3]. In PICSEL 3, the mediator integrates a local data warehouse which is progressively enriched by external data. More accurately, the main goals of the project include the Semantic treatment of the mediator answers and the studying of the reconciliation problem in order to eliminate redundancies and fusion data coming from different sources. Moreover, The SmartResource project [28] presents a general adaptation framework that adopts a two-stage transformation in order to represent the underlying XML-data in an RDF-based semantically rich format.

Compared to the above-mentioned approaches, the innovation of the hereby presented architecture relies on the flexibility and adaptability of a middleware in contrast to powerful, but case-specific approaches. The Priamos middleware provides an annotation mechanism that allows the abstraction of the outputs of each annotation layer. Trackers can be attached to a Priamos application that operate at the first (subsymbolic) layer, vocabularies can be chosen in order to provide the corresponding messaging interfaces (symbolic layer) and finally, rules can be defined that fuse and semantically enrich the information gathered, in order to provide a common Knowledge Base at a logical layer. Moreover, as analyzed in Section 3.1 and measured in Section 5, we propose an approach for the real-time processing of data, an aspect that was not investigated in any of the previous approaches.

From the context-aware point of view, there are systems related to the Priamos concept that use ontological descriptions to express contextual information. For instance, the Rei framework [29] uses RDF(S) or OWL Lite to represent context but the specification is limited to the terms of the Rei Ontology. We can also observe similarities between the work presented here and CoBrA [30]. Older approaches that investigated various aspects of the context-aware computing, like Ponder, the Context Toolkit [31], HP's CoolTown and the Intelligent Room project did not use a formal model to represent context information. The CHIL project[4] looks similar to the work presented in this paper as CHIL's goal is to control sensor data through an ontology based mechanism [32]. However, the description of the world model in CHIL is built/wired on the core vocabulary of the CHIL OWL ontology, not allowing the adoption of other ontologies targeting at different application scenarios, like in the hereby proposed system.

Semantic Web technologies specialized for ubiquitous computing have also been applied in several environments such as in the Task Computing environment [33], Gaia [34] and the SoaM Architecture [35]. However, unlike the aforementioned systems, the work presented in this paper focuses mostly in providing a middleware environment that does not restrict the users or developers to specific predefined vocabularies for a world model description or a message syntax among the various pluggable components. Emphasis is given in offering an architecture that is independent of ontologies and sensors while in the same time adopts a common formal representation of context and facilitates application development.

[3] The PICSEL project: `http://www.lri.fr/~sais/picsel3/`

[4] The CHIL project: `http://chil.server.de`

Commercial products that offer a complete RDF-based middleware have also made their appearence, such as Talis[5] or OpenLink[6]. Nevertheless, the purpose of this kind of middleware neither address real-time issues nor does it focus on context-aware multimedia applications. They are mostly focused on Web data.

The work here is also inspired by the one presented in [36] and moves on by supporting interoperability in various types of sensors as described in Section 7. The work in [36] is mostly focused on video analysis, which is also the case for VERL [4]. In the work presented here, video trackers are employed as a proof of concept and they could cleanly be substituted by e.g. speech detection trackers. As it is shown in the next Sections, Priamos offers an approach for bringing contextual multimedia annotations (metadata) to a higher level, by abstracting and splitting contextual information and annotation mechanisms.

3 Anatomy of the Middleware Architecture

This Section describes the infrastructure (see Figure 1) upon which solutions can be built that deal with situations where conventional content annotation is not sufficient and the need for Semantic annotation emerges. The middleware is designed with the aim to be configurable to any circumstances that can be modeled with the use of scenarios. The architecture comprises a set of core reusable distributed components for the real-time annotation of low-level context features and their mapping to higher-level semantics which are directly exposed through suitable APIs for application development. The main idea is to launch a procedure that annotates contextual multimedia information upon its appearance by using specific sets of rules. The resulting Knowledge Base reflects a spherical perception of the world model. An early description of the system was presented in [37].

First of all, the architecture abstracts the outputs from low-level, heterogeneous data sources (e.g. sensors, feature extraction algorithms, content repositories), thus enabling context capturing in varying conditions. Context annotation is configured through application-specific ontologies which can be plugged and it is automatically initiated without any further human intervention.

The basic components of the system are the data sources that are combined to low-level feature extraction components such as trackers, the user terminals running the applications, the administration console that handles the server and the Knowledge Base. Upon this infrastructure, intelligent applications can be built, communicating with the middleware via the Web Service API exposed. Figure 1 illustrates the different middleware components.

The trackers are the first ones to process raw data. Once initiated, they produce messages containing descriptions of the features captured from the sensors. Through these messages, knowledge is transferred to the main middleware server, it is converted into semantic information via rules, and a Knowledge Base is created where all the features of interest captured by the sensors are gathered. Section 3.1 presents a closer view into what goes on behind the scenes during the middleware operation.

[5] Talis homepage: `http://www.talis.com`

[6] OpenLink software homepage: `http://www.openlinksw.com`

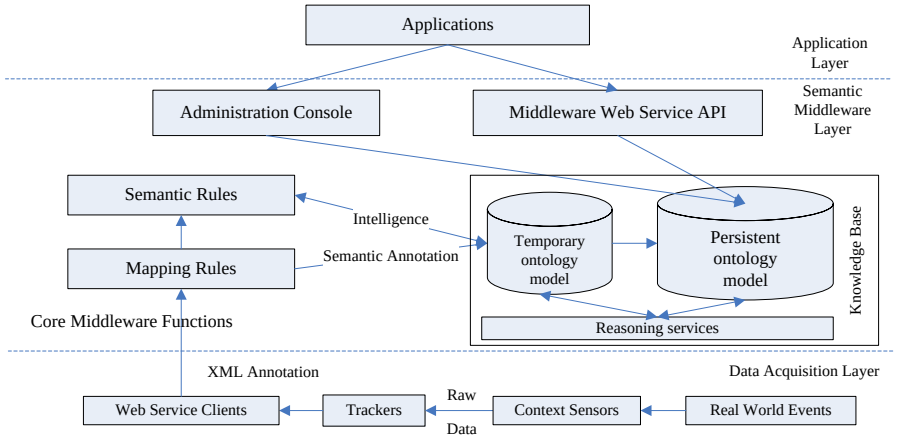

Fig. 1 The Middleware Architecture

3.1 Real-Time Message Processing

As depicted in Figure 2, the data are firstly aggregated and adapted, and then they are processed. Events that occur in real world comprise the raw data detected by sensors. Tracking algorithms are applied to these raw data and their results are sent via a Web Service message (technically an XML document enclosed in a SOAP envelope) to the middleware. When the message is received by the middleware, it is first checked for its validity. The only restriction for incoming messages is that they have to be in well-formed, valid XML format. The middleware poses no extra constraints. Firstly, the message is processed by the set of mapping rules. All of the mapping rules are applied to the incoming message. As a result of applying the mapping rules, the temporary Knowledge Base is updated with the new facts.

Consecutively, the set of the semantic rules is applied. This set of rules checks the conditions and performs actions related to the Database model, regardless of the contents of the XML message. This level of abstraction was chosen for two reasons; firstly because it separates XML mapping from semantic rules, facilitating the authoring process and secondly, because this processing phase can take advantage of the evolution of Semantic Web rule languages. Moreover, this approach decouples the annotation of the produced multimedia content with the desired application logic.

After the message process has terminated, the temporary ontology model has been updated. All added information is now stored in the ontology and what follows is the processing of the ontology itself. The rules are applied one by one keeping the model up-to-date with its context environment. New knowledge is potentially stored in the Knowledge Base after the arrival of each message. A technical analysis is presented in Sections 4.1 and 4.2 where two distinct scenarios are presented and the practical aspects are covered in more depth.

The maintenance that takes place in the last step of processing incoming messages can be configured according to the overall application. For the needs of our experiments, we considered two variables: the response time, which is the most important element of real-time systems, and the size of the ontology that directly affects the response time. The system can be configured with respect to either the response time or the size

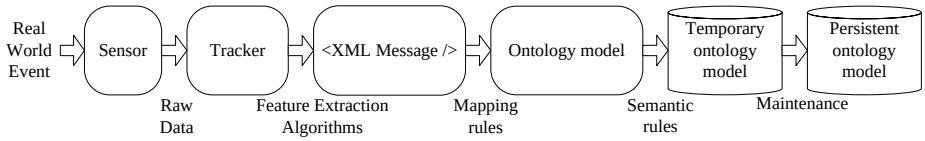

Fig. 2 Information flow: The set of the mapping rules is first applied to each incoming message. Then, the set of the semantic rules are applied. As a result, the ontology model is expanded and the new facts are added to it

of the ontology. The results exposed in Section 5 justify the real-time property of the middleware in terms of respecting the deadlines apposed and avoiding failure.

Technically speaking, the information flow initially begins as electromagnetic signals captured by a sensor. The tracker algorithm that runs on the sensor recognizes an event of interest, according to the kind of the sensor and the tracker. For instance, a camera (sensor) running the Viola-Jones algorithm (tracker) might recognize a human face in its domain of perception. This information has to be written in an XML file in accordance with an XML template that acts as an interface between the tracker and the middleware. This XML message is sent via Web Services to the middleware where the Mapping rules and Semantic rules are applied to it in order to store the new information in the system's Knowledge Base.

The main limitations of the approach are related to the real-time processing of the data. Theoretically, the system can be adjusted to any environment conditions. However, in practice, the user that composes the rules that define the system's behaviour is required to take extra care as conceptual mistakes can lead the Knowledge Base into growing out of bounds in short time. As an example, a rule that will trigger a new individual insertion after every incoming message will lead the ontology size to expand proportionally to the number of the incoming messages and will drastically reduce the real-time performance beyond acceptable times. The measurements that are presented in Section 5 are based on this rationale.

3.2 Software Modules

The modular software architecture mainly comprises an exported Web Service interface, the message templates, the ontology models, a set of mapping rules, a set of semantic rules, a set of available actions/notifications, the external reasoning server and finally the low-level components (e.g. sensors and trackers). This approach ensures its extensibility and adaptation to newer technologies. Every module is described analytically below.

Web Service interfacing module. The most important task of the Web service module is message manipulation. What is required from the messages that are sent from the Data Acquisition Layer (see Figure 1) is that they be expressed in any arbitrary well-formed XML document, without any additional constraints. When a message arrives, it is processed by the XML Mapping and Semantic Rules that the user has created in order to achieve the desirable behaviour. As far as the applications are concerned, the API contains functions to control the software modules and is described in better detail in Section 3.5.

Message Templates. The received messages can conform to any specifications we might choose. As previously mentioned, the only necessity is that the messages are well formed XML documents. Figure 3 shows a screenshot of the GUI offered by the administrative application through which the user can compose Mapping Rules that map paths in the XML tree of the message template to ontology classes. Information contained in the messages concerns environment elements such as person locations or time. Message templates are customizable according to the needs of each application. A message template can optionally have an attached DTD or XSD description.

An application can have several message templates, according to the messages that are expected to be sent by the trackers to the middleware. Different message templates can be chosen for different types of trackers. The structure of each message template is taken into account by the mapping rules applied to it, performing message separation. This separation enables message fusion at a semantic level with the use of Semantic rules.

Ontology models. An ontology model describing the classes, properties and their taxonomies is required for the system to work. The choice of an authoring environment is up to the user[7]. The ontology can be inserted to the middleware programmatically via the API exposed or via the Ontology Manager of the Administration Application (Section 3.3).

The ontology model is then stored using the Jena [39] internal graph engine. The Jena framework has developed its own methodology for storing and retrieving ontology information. In fact, the ontological model is stored in a relational Database in triples (statements of the form *Subject*, *Property*, *Object*) and form the underlying graph of the model. Jena allows many serializations of an ontology model, such as RDF/XML or N3 notation but the relational database backend is preferred in larger-scale projects because of several factors such as scalability or collaborative editing.

The annotation is kept in the Knowledge Base, separately from the incoming data that could be of any form, varying from simple text to A/V content and multimedia in general. Links to them are stored making possible a future retrieval.

In order to guarantee system's scalability, the ontology model that handles the incoming messages, i.e. the temporary ontology model, is kept in a different database from the persistent ontology model. The tradeoff for this approach is that the reasoning that takes place for every new message is aware of the facts that are stored in the temporary ontology model. Scheduled maintenance is responsible for migrating the facts from the temporary ontology to the persistent storage. This scheduled task can take place either synchronously or asynchronously. The former case indicates that the migration can be triggered by an incoming message while the latter that the migration schedule can be running as a daemon according to specified time intervals. Our approach follows the synchronous scheduling as shown in Figure 8.

We also must note that the use of several Semantic Web vocabularies is possible, desirable and encouraged in order to allow unambiguous definitions of the concepts involved in any application. Vocabularies such as Dublin Core Metadata Initiative for digital content[8], Creative Commons for licence information[9], or the Friend Of A

[7] According to [38] the most used ontology authoring environments are Protégé, SWOOP and OntoEdit.

[8] Dublin Core Metadata Element Set: `http://www.dublincore.org/documents/dces/`

[9] Creative Commons, Describing Copyright in RDF: `http://creativecommons.org/ns`

Friend (FOAF) network[10] provide the means for real semantic interoperability between applications.

Rules. Rules are essential in depicting the desired behaviour of context-aware systems. It is really convenient that the model-theoretic background of the OWL language [40] is based on Description Logics systems that are a subset of the First Order Predicate Logic. What is gained with the contextualization of a world model according to Description Logics is that first, the designed model has fully defined semantics and second, Horn-like clauses can be formed upon it. These clauses can be seen as rules that predefine the desired intelligence in the system's behaviour.

The Priamos middleware comprises two distinct sets of rules. The XML Mapping Rules (or simply Mapping Rules) and the Semantic Rules. The rules are formed according to the following event-condition-action (ECA [41]) pattern:

on *event* if *condition* then *action*

where the event in the Priamos case is the message arrival. The Mapping Rules fetch data from the XML message and store it into the ontology model in the form of class individuals. They rely on the fact that for every XML element there is a unique XPath[11] expression that retrieves its value.

The Semantic Rules on the other hand perform modifications on the ontology model. Therefore, despite the common syntax, the conditions and actions differ in each case as it is shown in Appendices A and B. The distinction of two variants of rules was inspired by the similar distinction between the RDF-only and the RDF-XML-combining subsets of RuleML [42] [43]. In our case, this distinction allows the processing of the messages in two steps. First, by taking into account specific XML elements in each incoming message, the middleware is able to separate the messages from various trackers. Second, the incoming information can be fused with the use of Semantic rules. Detailed examples are provided in Section 4, where two test-case scenarios are presented in more detail.

The Reasoning Server. As stated in [44], a Knowledge Base is the combination of an ontology and a reasoner, thus the presence of the latter is indispensable. There is a variety of available reasoners, commercial like RacerPro or OntoBroker, free of charge like KAON2 [45] and open-source like Pellet [46] and FaCT++ [47]. All of them support DIG [48] interoperability which is not a standard yet but it is used by reasoners to exchange HTTP messages with programs that call them. Jena also supports the binding of an external reasoner, and provides a less adequate internal reasoner as well. The previously mentioned reasoners can function as stand-alone DIG servers and communicate with the Priamos middleware, leaving the reasoner choice up to the user. Figure 7(a) demonstrates the results we obtained by using Pellet, FaCT++ or no reasoner at all during system's operation.

Trackers. The trackers are the first ones to process raw data. They apply special algorithms and techniques to the signal captured by the sensors (e.g. object/human detection, face detection, recognition and tracking, audio localization) in order to identify features of interest. Once initiated, the trackers produce XML messages that describe their awareness of the world. Through these messages, knowledge is transferred via Web Services to the main Priamos server. Thus, any tracker, able to produce a Web Service message is pluggable in the middleware architecture. For the scenarios described in

[10] FOAF Vocabulary Specification: `http://xmlns.com/foaf/spec/`

[11] XPath 2.0 is a W3C recommendation since Jan. 23, 2007: `http://www.w3.org/TR/xpath20/`

Section 4, the Viola-Jones [10] algorithm for face detection and Camshift [15] for face tracking were used.

3.3 Administration Application Description

The tools that are used to handle the various components are developed in Web environment and they consist of: the trackers, the ontology manager, the message template manager, the action manager, the message to ontology mapper and finally, the Semantic rule composition mechanism.

Ontology manager. The user can upload models to the system and store them in two forms: in plain text in the database, and using Jena's persistent storage engine. The ontology text is stored in the database for portability purposes; it could as well be stored in a plain text file. The user decides which description language he should use: RDF(S) [49] or OWL [50] [40] [51]. The system's ontologies have no theoretical limitation in description and evolution. The only profound limitation is that using the OWL Full variant of the OWL language, will not be supported by a reasoner. In any other case, consistency will be guaranteed by the reasoner. We also note that ontologies, today, are easy to find on the Web[12] and usually it is more convenient to customize an ontology according to an application's needs than to start authoring from scratch.

Message template manager. The messages that can be received are stored in the database because they are needed during the mapping process. The user can add and delete message templates. Validation is carried out during the insertion to ensure future unimpeded function.

Action manager. We offer control of the actions that may be triggered while the Semantic Rules are processed. At the moment four types of actions have been implemented whereas new application-specific actions can be added through the administration application or the middleware API.

- Send SMS: The Priamos middleware can send an SMS to the mobile phone of the users when an important event takes place
- Send email: According to the events handling, the users can also be informed by email
- Send Web Service message: The Priamos middleware can call an external Web Service
- Run an application: An external application can be executed either locally or remotely.

Message to ontology mapper. We have developed a mapping language to allow the composition of rules that will bind each message to the classes of an ontology. The developer can assign Mapping Rules to specific models. These rules will be processed one by one upon the arrival of each message and they are responsible for adding the extra information in the ontology. The Mapping Rules follow the rule grammar described in Appendix A.

An example of a Mapping Rule can order the system to check whether a specific element exists in an incoming message or not. If the check is successful, then the rule commands the system to insert an individual to a certain class in the ontology. For example, the rule: `if XPath exists then insert individual in OntClass`, can be

[12] Among the most reliable sources is the prominent Swoogle (`http://swoogle.umbc.edu`). Noticeable results are also produced with the `filetype:owl` or `filetype:rdf` google operators

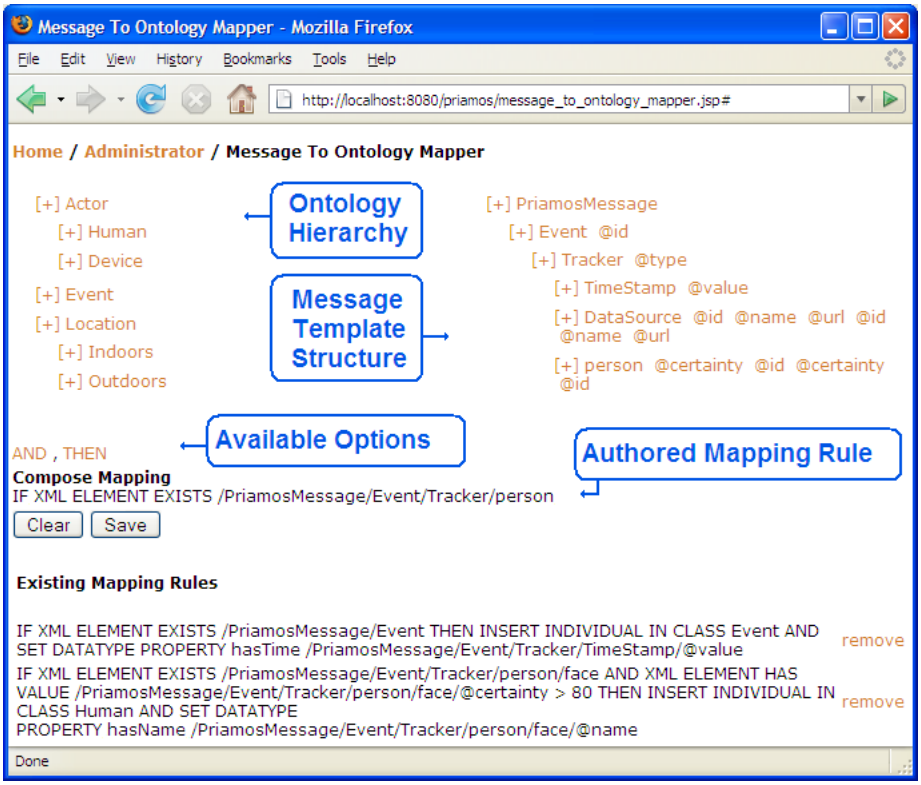

Fig. 3 Mapping rules authoring. The user selects paths in the XML tree of the message template and authors rules that map these paths to concepts of the ontology hierarchy

used to insert an individual in class `Person` if the path `Message/Event/Person` exists in the message. The Mapping rule composer, part of the Administration application, displays the ontology hierarchy on the left, the XML tree on the right and the defining rules underneath, as shown in Figure 3. The composer allows the user to graphically define and process the Mapping rules of the application built on top of the Priamos middleware.

Semantic Rule composition. The application developer can define rules that are processed on the model. The developer does not have to be a domain expert or have specific knowledge of the underlying infrastructure. The Semantic Rules follow the rule grammar described in the Appendix. An example of a rule that can be declared is: `if OntClass has individuals then Alert (Message)`. In this case, the system will call a predefined action named Alert (i.e. an SMS, an email, a Web Service message or an external command) if the check for individuals in a class e.g. `DangerousEvent` returns true. A graphical authoring tool is provided based on this rationale for the composition of rules by non-expert users. The Semantic Rule composer component of the administration application displays the ontology hierarchy on the left and the defining rules underneath, in a way similar to the Message to ontology mapper shown in Figure 3.

3.4 Users and Roles in Priamos

The Priamos core functions facilitate application development, in different scenarios and context configurations. Users that benefit from the Priamos technology are classified into two categories: developers and end users.

Developers. They have the responsibility of defining the Mapping Rules from the incoming messages to the ontology concepts. Instead of developing application specific code each time, the developers can exploit the core middleware functionality. They can "plug" an ontology, form Semantic Rules on the ontology, and define the actions that can be taken. They have the freedom to tune the system's behaviour as wished, through Priamos provided event handlers and callback functions.

End users. They have the overall supervision of the system's functions and can configure it for different operation scenarios. The end users can define features of interest to be captured (e.g. when a security alert should be triggered). They can be simply monitoring a system operation session, or waiting to receive automated notifications in form of a sound, an email, a call, an alert in general (e.g. a security guard in a security-surveillance scenario who receives alerts in his mobile). No expertise or knowledge of the system's underlying infrastructure is required from the end users.

3.5 The Priamos API

In order to make Priamos middleware easily adopted to external applications we have created a set of functions that a developer can call. The middleware exports a Web Service API through which an application can model its logic. For instance, through the Priamos API we can have the middleware setup in a single application. No further actions are needed than a script execution in order to have the middleware up-and-running. A summarization of the Priamos API functions is shown in Table 1.

Message Templates	Semantic Queries	Turn On/Off Middleware
`insertMessageTemplate`	`getQueryNames`	`turnOnMiddleware`
`removeMessageTemplate`	`insertQuery`	`turnOffMiddleware`
`getMessageTemplateId`	`removeQuery`	`getMiddlewareStatus`
`getMessageTemplateNames`	`getQueryContents`	
Ontology Models	**Rules**	**Alerts**
`insertOntology`	`insertMappingRule`	`insertAction`
`removeOntology`	`insertRule`	`removeAction`
`getOntologyId`	`removeMappingRule`	`getActionPathToExecutable`
`getOntologyNames`	`removeRule`	`getActionNames`
`getOntologyContents`	`getMappingRules`	
	`getRules`	
Incoming Messages		
`priamosMessage`		

Table 1 A short summary of the exposed Web Service Priamos API functions

The methods offered by the Priamos API include manipulation of the ontology models, message templates, rules and actions of the application, along with general-purpose functions such as to turn on and off the middleware message processing. They

can be called during the different phases of its operation. The rationale behind implementing a Web Service API is that the technology independent Web Service allows ease of integration of trackers or applications regardless of the platform of their implementation.

4 Test Case Scenario: Priamos in Action

Great effort has been put into creating, managing and administering Semantic information. Nevertheless, it is common belief that the Semantic Web is not yet established because of lack of applications that exploit the Semantic information. In order for the semantic information to be more widespread, it should be clear how it is useful and how the end user benefits from its existence. This Section demonstrates and analyzes two scenarios, two applications that are based on the middleware described in the previous Section. The purpose of the hereby analysis is the justification of the claim that adoption of Semantic Web technologies brings actual intelligence to the application.

Subject	Property	Object
default:Actor	rdf:type	owl:Class ;
	owl:disjointWith	default:Event , default:Location .
default:Known	rdf:type	owl:Class ;
	rdfs:subClassOf	default:Actor .
default:Unknown	rdf:type	owl:Class ;
	rdfs:subClassOf	default:Actor .
default:Professor	rdf:type	owl:Class ;
	rdfs:subClassOf	default:Staff .
default:Staff	rdf:type	owl:Class ;
	rdfs:subClassOf	default:Known .
default:Student	rdf:type	owl:Class ;
	rdfs:subClassOf	default:Known .
default:Undergraduate	rdf:type	owl:Class ;
	rdfs:subClassOf	default:Student .
default:Postgraduate	rdf:type	owl:Class ;
	rdfs:subClassOf	default:Student .
default:hasXLocation	rdf:type	owl:DatatypeProperty ;
	rdfs:domain	default:Actor ;
	rdfs:range	xsd:int .
default:hasYLocation	rdf:type	owl:DatatypeProperty ;
	rdfs:domain	default:Actor ;
	rdfs:range	xsd:int .
default:hasCertainty	rdf:type	owl:DatatypeProperty ;
	rdfs:domain	default:Actor ;
	rdfs:range	xsd:int .

Table 2 The ontology that supports the Security surveillance scenario

4.1 Security surveillance scenario

In this Section, we describe a smart security surveillance scenario based on the proposed architecture and the exposed API. The scope of the scenario is to better clarify the

functionality and the benefits of the use of the middleware. In this use-case scenario, an application is built on top of the Priamos middleware and it is used to monitor a room and request alerts in case different events take place. The environment consists of a series of cameras, the Priamos middleware and the Surveillance application[13].

Firstly, we present in Table 2 a fraction of the ontology that supports the surveillance scenario. Ontology authoring was inspired from [52] where the authors adopt an approach based on the concepts of *resource*, *actor*, and *environment* in order to describe a context model.

In short, the class of importance for the scenario is the class `Actor` that has two subclasses: `Known` and `Unknown`. Class `Known` has two subclasses `Staff` and `Student`. `Staff` has only one subclass, `Professor`, and `Student` has two subclasses: `Undergraduate` and `Postgraduate`. The datatype properties `hasXLocation`, `hasYLocation` and `hasCertainty` apply to all of these classes.

Firstly, once the system is activated it detects automatically the type of devices that are connected to it (i.e. cameras and microphones). Then an end user connects online to the control panel of the middleware and checks the trackers that it recognizes, in our case two trackers for movement recognition. The trackers return messages in XML format, e.g. the Body Tracker recognizing a human body in its visual range returns coordinates. Note that the message retains a reference to the url of the original multimedia file as the value of the `url` property.

```
<Event id="5712">
  <Tracker type="FaceTracker">
    <DataSource id="3" name="CeilingCamera" url="http://localhost/seq_0077.jpg"/>
    <person id="1" certainty="100">
      <location2d datasourceId="3" x="429" y="46"/>
      <face dbpersonid="10" name="John" certainty="91"/>
    </person>
  </Tracker>
</Event>
```

For our example purposes, we incorporate a Face Tracker that implements two algorithms; the Viola-Jones algorithm for face detection [10] and the Camshift [15] algorithm for face tracking. The face detection algorithm is preconfigured to recognize a fixed set of persons. Camera input is analyzed in real-time, producing the rectangle areas in the screenshots in Figure 4 (and later, in Figure 6) and XML messages of the previously discussed form.

The first event that the system captures is when a new person (a new face) appears on screen. For our surveillance scenario we first check the persons entering the room. A Mapping rule that checks the incoming messages for persons is the following:

```
if,xml element has value,/Event/Tracker/person/face/@dbpersonid,eq,10,then,
insert individual in class,Professor,
and set datatype property,hasXLocation,/Event/Tracker/person/location2d/@x
```

This rule checks the incoming XML message and, if a certain condition is met (i.e. a specific path in the message to have a specific value), a new person will be inserted in the Knowledge Base. However, in real world conditions, the rules will be more complicated as we would also modify several other datatype properties of the

[13] Not to be confused with external software applications, here by the term "application", we note an application built on top of the middleware, i.e. the middleware configured for specific sensors, ontology models and message syntax.

(a) (b) (c)

Fig. 4 In (a) the Viola-Jones algorithm detects a face of a person that enters the room. In (b) and (c) we see the Camshift algorithm tracking two faces

individual such as his coordinates and object properties such as the event in which the individual participates. Moreover, in order to deal with possible inconsistence in the outputs of the trackers, we would apply a certainty threshold (e.g. accept an event only if the tracker gives a certainty greater than 90 percent). As an example of a more concise rule that satisfies these requirements, we would state:

```
if,xml element exists,/Event/Tracker/person,and
xml element has value,/Event/Tracker/person/face/@dbpersonid,eq,10,and,
xml element has value,/Event/Tracker/person/face/@certainty,gt,90,
then,
insert individual in class,Professor,
named after,/Event/Tracker/person/face/@name,
and set datatype property,hasXLocation,/Event/Tracker/person/location2d/@x,
and set datatype property,hasYLocation,/Event/Tracker/person/location2d/@y,
and set datatype property,hasCertainty,/Event/Tracker/person/face/@certainty
```

Similarly, the system can be trained to recognize the persons that the face tracker recognizes, simply by mapping their id (10 in the example above) to the corresponding class in the ontology. This rule will lead to the insertion in the ontology of a triple of the following form if the person is recognized e.g. as a Professor:

```
default:John       rdf:type              default:Professor ;
                   default:hasXLocation  "429"^^xsd:int .
```

If the tracker finds a person that is not recognized, the triple that will be added to the model will have a form similar to the following:

```
default:Unknown_1  rdf:type              default:Unknown ;
                   default:hasXLocation  "429"^^xsd:int .
```

Another rule could insert a person that the system recognizes to the class `Student`, etc. so that the system can be adjusted according to our knowledge of the world.

In continuation, the Semantic rules are executed. Given the abundant Priamos rule language (fully described in Appendices A and B) several precautions have been declared. The following rules are Semantic rules, and they are responsible for the alerts produced by the middleware, based solely on the ontology's state.

i) It is not allowed to a Student to enter or stay in the room without the attendance of a Staff member. This rule can be expanded in other scenarios; the use of classes `Staff` and `Student` is not restrictive.

```
if,class has individuals,Unknown,and,
SPARQL query does not have results,
SELECT ?x ?y WHERE {
    ?x rdf:type default:Staff .
    ?x default:hasTime ?time1 .
    ?y rdf:type default:Student .
    ?y default:hasTime ?time2
FILTER (?time1 = ?time2) }
then,
alert ("Unknown unattended person detected")
```

ii) During working hours only known persons are allowed.

```
if,SPARQL query has results,
SELECT ?x ?y WHERE {
    ?x rdf:type default:Unknown .
    ?x :hasTime ?y .
FILTER (?y > "20:00:00"^^xsd:time || ?y < "08:00:00"^^xsd:time) }
then,
alert ("Unknown person detected during working hours")
```

iii) When high-security level time (e.g. after 23:00 and before 7:00) nobody should enter the room.

```
if,datatype property in class Actor,gt,"23:00:00",and,
datatype property in class,Actor,lt,"07:00:00",
then,
alert ("Person Detected during High-Security Level Time")
```

In the above cases, `alert` is a command that might turn on the lights, set an alarm, send an email, an SMS or a Web Service message. For example if a person is detected in the room under surveillance before 7:00 in the morning, a guard can be notified by an alarm and a SMS can be sent to his mobile.

The benefits of the current approach can be summed up in the use of intelligence, the expressiveness in defining the desired system's behaviour while at the same time dealing with the real-time processing of the incoming bulk of metadata. In addition, a searchable Knowledge Base is offered for future exploitation.

The reason that justifies the use of Semantic Web technologies is that firstly, the created system can be adjusted easily to different scenarios and thus presents increased reusability. For rule (i) for instance, instead of using classes `Staff` and `Student` we could have classes `Adult` and `Minor` for another scenario, or any two classes, `Class_A` and `Class_B` whose parallel existence we wish.

Second, a simple rule that checks the existence of a `Student` (i.e. the statement `if,class,has individuals,Student`), combined with the reasoning procedures will calculate among the results individuals of the classes `Postgraduate` and `Undergraduate`, eliminating the need for declaring extra rules. The same holds for the class `Staff` in which are also counted the individuals of its subclasses. Thus, the rule (i) captures a series of combinations of coexisting individuals from classes that are connected between them with super/subclass relationship.

We are mostly relying on the super/subclass relationship of the model in the scope of the examples but the use of a reasoner unleashes all the OWL DL potential in world description.

4.2 Smart Meeting Room

Additional testing was conducted under the terms of a different scenario, the scenario of a smart meeting room. The scenario's purpose is to investigate the ease of the system's adoption under different and more complex system's operational requirements. In this scenario, a room is monitored, in which meetings take place. All of the persons are waited to be present before the meeting starts. During the meeting, one person goes to the presentation area and starts his/her presentation while the rest of the persons are sitting. After the end of the presentation, the person sits down with the rest of the participants and continue the meeting. The meeting ends when all of the participants leave the room. In this scenario, we can notice four distinct system states, which are depicted in Figure 5.

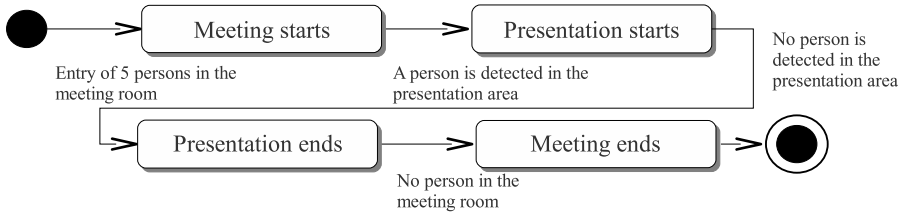

Fig. 5 State diagram of the Smart Meeting Room scenario

The sensors and trackers of the Security surveillance scenario are employed here as well, with the addition of a move detection tracker that is connected to a panoramic camera placed on the ceiling of the room. Hence, `x` and `y` coordinates in this scenario refer to the persons' locations in the room.

The ontology that supports this scenario is similar to and extends the ontology that supports the Security surveillance scenario (Table 2). The only addition necessary is a class `State` with a datatype property `hasState`, that holds information about the system's state. The rules that support this scenario are the following:

A **Mapping rule.** The Mapping rule that transforms the incoming XML messages into semantically enriched information is similar to the Mapping Rule presented in the Security Surveillance scenario.

```
if,xml element has value,/Event/Tracker/@type,eq,"PanoramicTracker",and,
xml element exists,/Event/Tracker/person/@id,then,
insert individual in class,Person,
named after,/Event/Tracker/person/@id,and
set datatype property,hasYLocation,/Event/Tracker/person/location2d/@y,and
set datatype property,hasXLocation,/Event/Tracker/person/location2d/@x,and
set datatype property,hasTime,/Event/Tracker/TimeStamp/@value"
```

Eight **Semantic rules** (four pairs). These rules define the transitions between the system's states and the middleware's actions. In fact these rules implement the model in Figure 5. Below follows an example for the transition from "Meeting_Started" state to "Presentation_Started" state. The numbers in the rules define the coordinates of the presentation area.

```
if,datatype property in class,System,hasState,eq,Meeting_Started,and,
```

```
datatype property in class,Actor,hasXLocation,gt,400,and,
datatype property in class,Actor,hasYLocation,gt,300,then,
send web service message,"Presentation Started!"

if,datatype property in class,System,hasState,eq,Meeting_Started,and,
datatype property in class,Actor,hasXLocation,gt,400,and,
datatype property in class,Actor,hasYLocation,gt,300,then,
execute SPARQL query,
PREFIX default: <http://www.example.org/meeting.owl#>
DELETE { ?x default:hasState ?z }
WHERE  { ?x default:hasState ?z }
INSERT { default:System_State a:hasState 'Presentation_Started'}
```

As a result, besides the real-time alerts that the system can provide, the Knowledge Base that is created can answer to queries such as:

- When did a certain person make a presentation?
- When was the last time a guest (an unknown person) made a presentation?

The application is of higher complexity because of the larger set of Semantic rules and the additional burden is reflected in the measurements in Figure 8.

5 Performance Evaluation

The Priamos middleware was thoroughly tested about its scalability. While measurements were taken using a Linux server, based on an Intel Core 2 Quad at 2.40 GHz, 3 GB RAM, the relations among different sets of test are indicative and independent of specific computing system composition.

The environment of our tests consisted of a camera, connected to a computer running the Viola-Jones algorithm for face recognition and the Camshift algorithm for face tracking. The middleware was connected to a camera producing XML messages that were sent through a Web Service to the computer running a preconfigured version of the middleware. The system was configured so that it would capture faces from the camera's visual range, track them and send information to the server. In continuation, the server populated a given ontology with individuals and store information about the individuals' location and time. The reasoner was bound to the underlying model of the ontology for each incoming message and assertional information was deduced using combinations of Mapping and Semantic Rules.

Our data sets included various versions of trackers that performed operations on video surveillance results. In Figure 6, it can be seen that erroneously, early versions of the face tracker recognized persons in the video, providing us with a number of persons that would otherwise be unavailable. Thus, in the datasets, "faces" appear in certain locations, are being followed by the face tracker and disappear in later scenes. This provides videos in which a large number of "faces" are born and lost, simulating the system's operation on active, lively environments.

In Figures 7 and 8 the axes represent the *Process time* in function of the incoming messages. As *Process time*, is considered the time spent from the arrival of a Web Service message from a tracker till its storage in the Knowledge Base. In other words, the process time is the time it takes for both Mapping rules and Semantic rules to be executed upon the arrival of an XML message. The *Incoming messages* represent the number of the messages that have been received from the middleware. The running

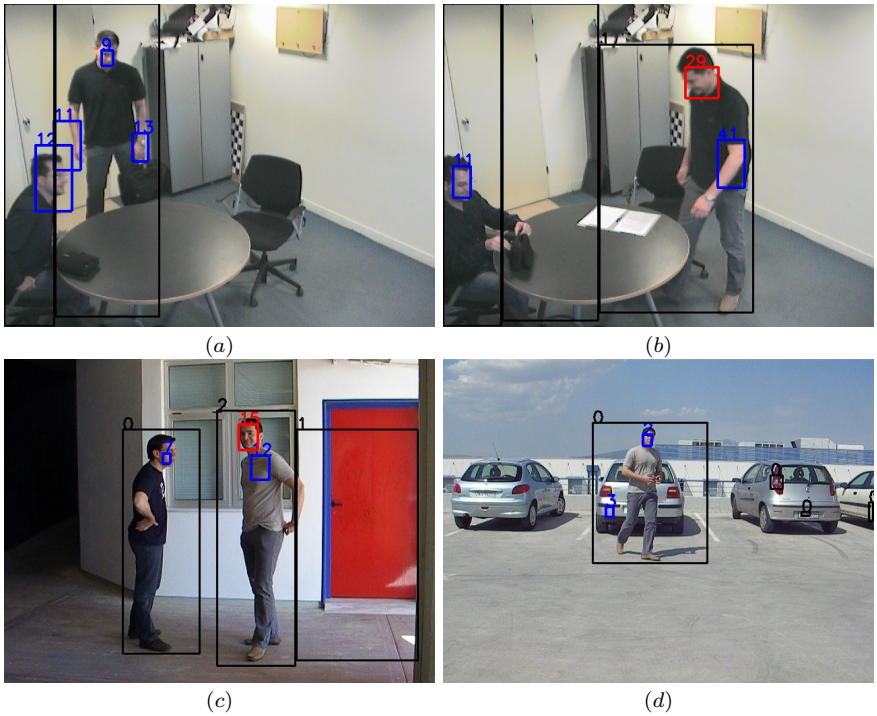

Fig. 6 Instances of data used for measurements. It can be clearly seen in (b) that because of an incomplete and early version of the trackers, the system has already recognized 41 faces while actually only 2 entered the room. This provided the input for our measurements

average in the graphs constitutes the average process time based on the latest 50 messages for any given message. Its purpose is to normalize the fluctuation in the process time of each message and give a clearer picture on the results (see also Figure 10(a) and 10(b)).

Figure 7(a) demonstrates the results of testing the system's behaviour under the same conditions with the only difference in the reasoner server employed. The reasoners tested were Pellet and FaCT++ and, as shown in the figure, they produced similar results with the former being slightly faster.

Figure 7(b) demonstrates the system's behaviour under three conditions. The Mapping rules are "dummy" rules that insert a new individual in an ontology class for each newly arrived message causing the size of the ontology – and specifically, the ontology's ABox – to explode. These measurements were conducted with the first versions of the middleware and they led us to reckon that the need of a synchronous maintenance method as shown in Figure 8 is indispensable.

Meanwhile, we can observe in Figure 7 (a) and (b) that the absence of a reasoner reduces the time needed for processing each incoming message. This is justified by the fact that no inference needs to be performed on the ontology and thus a big calculation burden is avoided. The tradeoff in that case is a loss in the accuracy of processing the rules. For instance, a rule atom stating "`if class has individuals`" will wrongly return *false* if the individuals are not direct individuals of the class and belong to one

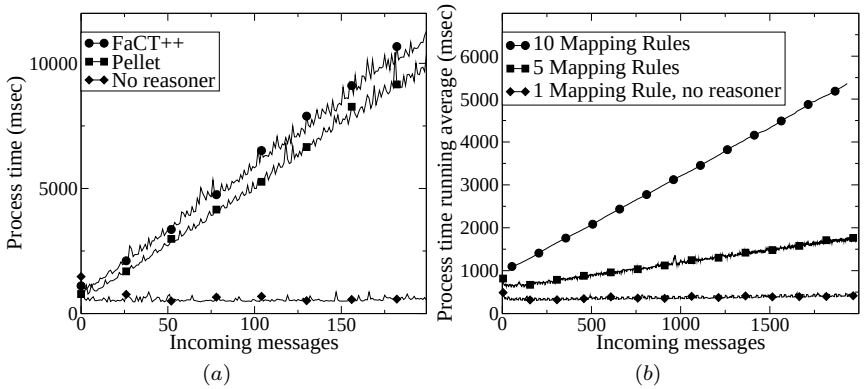

Fig. 7 In (a) a comparison of reasoning servers is presented and from (b) we can deduce that the processing time depends on the size and complexity of the rule sets

of the subclasses, since they will not be discovered at all. Absence of reasoning services is a great boost for performance. In the experiments conducted, we noticed a decrease of system latency. However, the lack of reasoner support means that no intelligence is added to the system. Thus, the absence of a reasoner is discouraged and it is suggested only in cases that knowledge extraction is a minor feature.

Inevitably, the use of reasoner causes the ontology model to increase leading to a further increase of the latency of the system. The proposed solution to that problem is the synchronous maintenance operation on the ontology model, configured to alleviate ontology's load. The saw-like graphs shown in Figure 8 were created with the use of synchronous process of the incoming data. The thresholds in each case were chosen in order for the resulting graphs to be competently illustrative of the application's behaviour.

In Figure 8(a), the maintenance method used aims at keeping latency under the threshold of two seconds while Figure 8(b) performs maintenance when the number of the triples in the ontology model exceeds 180. In both cases, when the threshold is reached, the ABox of the ontology (the individuals) are moved from the temporary ontology model to the persistent ontology with the use of the SPARQL/Update[14] language [54]. Both methods demonstrate similar results. It can be seen from the results in Figure 8(c) and (d) that the meeting room presents bigger response times than those in the simple surveillance scenario, a fact which is due to the size and complexity of the rule sets.

In practice, lower thresholds in either the process time or the size of the ontology restrict the system's awareness of the world. This happens because for the real-time processing of the incoming messages, the system's perception of the world is bounded to the temporary ontology model. As shown in the graphs, the thresholds that define the "capacity" of the temporary ontology model should be kept high enough in order for maintenance operations to take place in adequately sparse time intervals.

In order to evaluate more objectively the performance of the middleware in terms of speed, we conducted a series of measurements out of the scope of the two scenarios previously presented. The measurements below were taken on a set of offline data, a

[14] Delete or Update functions are not included in the W3C SPARQL recommendation [53].

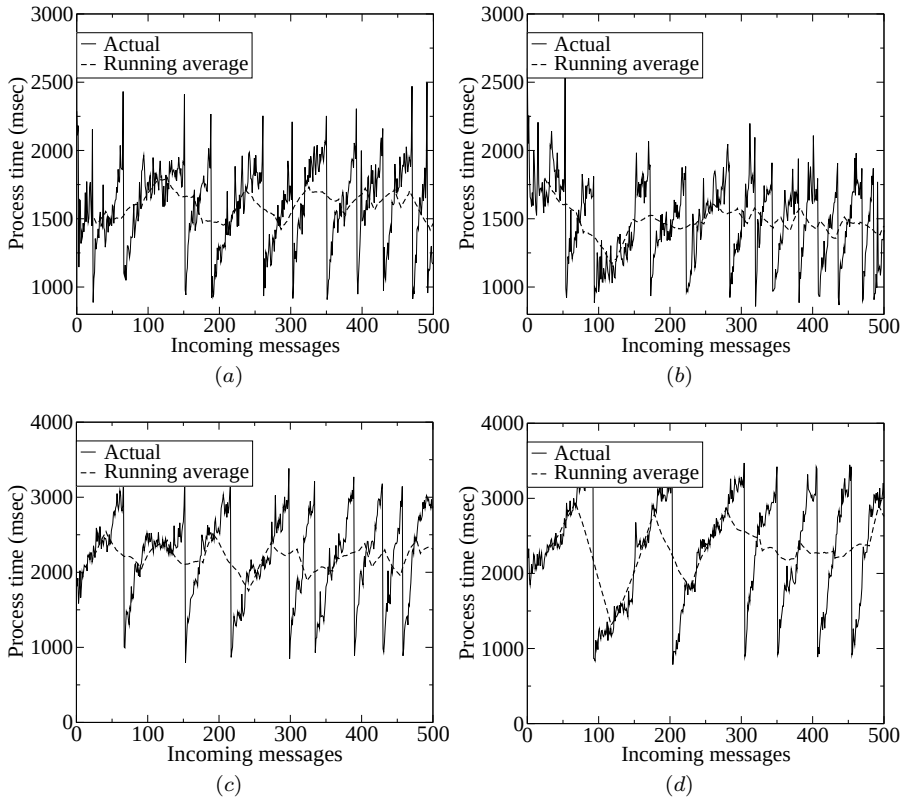

Fig. 8 In (a) and (b) we can see the behaviour of the Surveillance monitoring application and in (c) and (d), the Smart meeting room application. Reasoning support is enabled and latency is maintained at certain threshold. In (a), the response time is kept under 2", in (c) under 3", while in (b) the ontology triple entries are kept under 180 and in (d) under 80

series of XML files containing information captured by cameras, processed by a face detection/recognition tracker and a face tracker.

Among the most important observations is that the threshold that the user can set directly affects the time intervals between maintenance operations. In the two experiments whose results are displayed in Figure 9(a), every other parameter was kept the same, except the threshold, that was set in 1600 and 800 milliseconds respectively. We notice that higher thresholds (time thresholds in these cases) lead to less frequent maintenance operations. Also, we observe that a more dense flow of incoming messages (e.g. from more or faster trackers) will shrink the time intervals between maintenance operations in Figure 8. Poor choices in rule composition, ontology model, reasoner and/or RDBMS will shift up the graphs. Reversely, optimization in each component's properties will cause the graphs to expand horizontally and/or shift down. The important observation is that under any circumstances the graphs retain their designative saw-like form, with the process time always restrained under the imposed threshold.

The next experiment, as depicted in Figure 9 (b) and (c), demonstrates that the process time is directly proportional to the triple size of the ontology in the temporary

Knowledge Base. Due to this fact, the threshold for each application can be arbitrarily chosen to be either in milliseconds or in triples. However, as it can be seen in Figure 9(c), the performance of an application will be improved for relatively large ontology sizes. In other words, a relatively low threshold (below approximately a 100 triples in this experiment) is not profitable for the performance of an application as a high Process time/triple will appear.

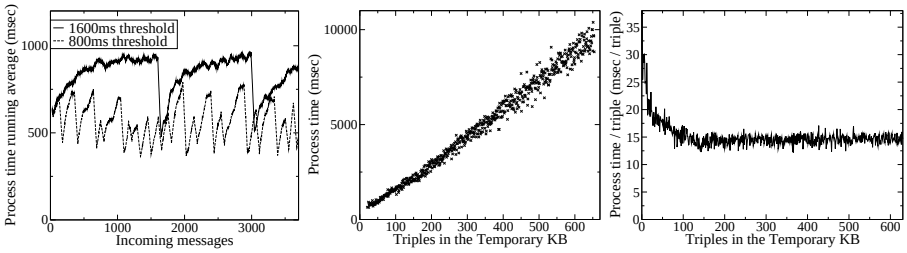

Fig. 9 In (a) we can see that higher thresholds lead to less frequent maintenance operations. In (b) it is demonstrated that the process time of each incoming message is directly proportional to the size in triples of the temporary Knowledge Base. In (c) we can see that the process time per triple in the temporary Knowledge Base is high for relatively small ontology sizes, but it is stabilized for larger ontologies

Figure 10 demonstrates the results of the next experiment: the behaviour of a single tracker plugged in the middleware. The threshold is set to 160 triples. Figure 10(a) depicts the actual time in which the client receives an acknowledgement from the middleware that each message sent was processed correctly. In 10(b) and 10(c), the running averages from the latest 50 messages are displayed (as in Figure 8) in order to demonstrate the similarity in the client and the server response times, normalizing any small fluctuations. As expected, the behaviour is identical through time. The only latency overhead is caused by the network, as shown in Figure 10(e). This overhead remains steady at approximately 100 milliseconds and it can be considered insignificant. We must also mention that the measurements agree with our previous observation: the process time of the server for each message is directly proportional to the number of the triples in the temporary Knowledge Base, as it is shown in 10(d).

With three clients operating concurrently over the (same) data, the measurements demonstrate a more interesting fact. Figure 11 depicts the behaviour of a Priamos application with three clients operating concurrently on a set of offline data. Each one of them sends 3000 messages to the middleware server. Graphs (a), (b), and (c) in Figure 11 depict the respective client behaviour, while (d) and (e) depict the server behaviour and the number of triples in the server's temporary Knowledge Base, as well. The server processes a total of 9000 messages, 3000 from each client. Note that each client's behaviour through time is similar to the server's. However, as we would expect, the response time for each client – i.e. the time needed to receive acknowledgement from the server – is greater than the time needed by the server to process one message, a fact due to message synchronization in the server. Therefore, the performance of a Priamos application is substantially influenced by the number of trackers operating concurrently.

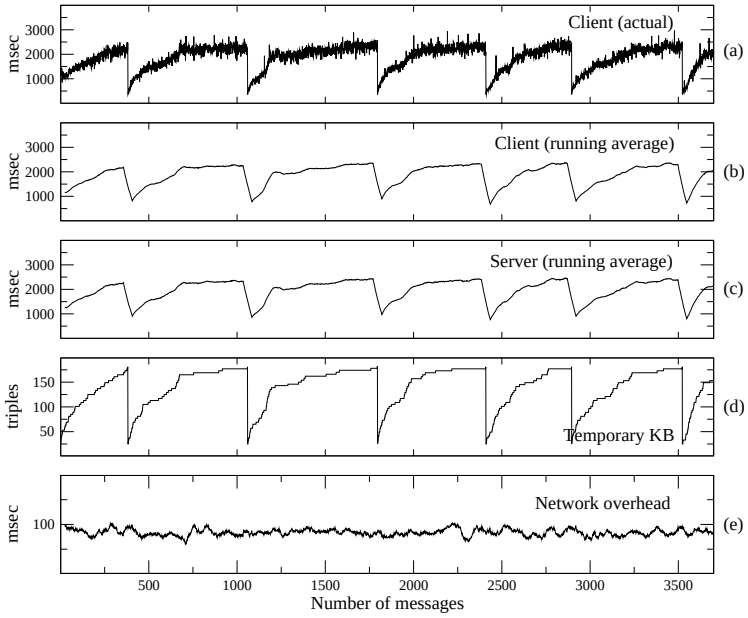

Fig. 10 A client processing offline data. We can observe that the client's performance is similar to the server's. The performance is mainly affected by the triples in the temporary Knowledge Base, while the network overhead remains relatively unimportant

Having analyzed so far the performance in terms of speed, we must also note that an equally important performance property is *accuracy* in the annotation of the multimedia data. In other words, the choice of trackers is crucial in order to present adequate statistical properties. An ideal tracker – for face, speech or event recognition, for instance – will have a very high true positive rate (also referred to as detection rate) and a very low false positive rate. As a consequence, in terms of accuracy, the system's behaviour is defined by the attached trackers. In the Priamos case, the face recognition follows the approach described in [55]: A sub-class Linear Discriminant Analysis approach is used for feature extraction and a Nearest Neighbor classifier for obtaining the identity per face. These identities are fused for all faces in the track using a weighted voting scheme. Further augmenting the feature extraction part with DCT normalization increases recognition performance by 4 percent, surpassing 90 percent correct recognition rate given just one second of video for fusing the decisions. Regarding face tracking, the Camshift-based approach that is followed is presented and evaluated in [56].

Now, regarding possible failures in any of the components that can affect performance, we can consider two distinct cases. First, a tracker can fail to create a message. In this case the message does not reach the middleware. Second, a tracker can create a false positive. In this case, a message arrives at the middleware. However, the mistake can still be dealt with by appropriate Semantic rules applied in order to assure integrity in the values conveyed by the messages. For instance, a face that suddenly appears and suddenly disappears and does not present continuity in its trajectory can be discarded

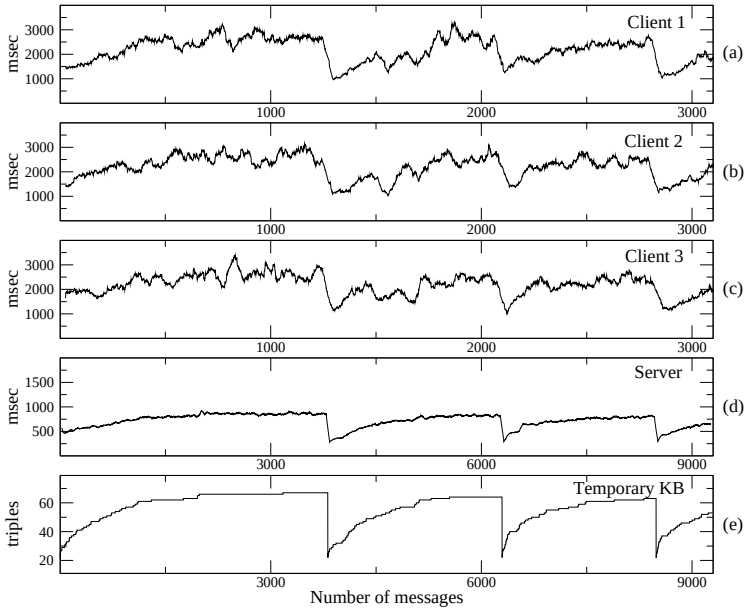

Fig. 11 Three clients operating concurrently over pre-processed offline data. A three-to-one ratio appears also at the respective performance

with the use of Semantic rules during message fusion and hence, the number of false positives can be furthermore decreased.

If a tracker produces overwhelmingly many messages, the saw-like graphs in Figure 8 will be more frequent, meaning that the middleware will perform more frequent maintenance operations. An approach to dealing with such a case is demonstrated in Figure 9(a), where a higher threshold leads to less frequent maintenances. However, under any circumstances, the behaviour of the middleware is characterized by the notion of hard real-time: the deadlines imposed are always met.

6 Conclusions

The work presented in this paper aims at providing an approach for the real-time Semantic annotation of contextual information for enabling the development of innovative intelligent applications. We presented Priamos, a Semantic Web standards-based middleware architecture for semantically annotating rich context information in real-time. A distinguishing characteristic of Priamos middleware is that it separates the content annotation functionality from the application logic, which can be plugged into the middleware architecture in the form of an ontology and can be easily processed by the application developer through the APIs that Priamos provides. Thus, the development effort is eliminated enabling a variety of applications featuring real-time alerts, such as in smart meeting rooms, advanced surveillance, environmental monitoring etc.

Another innovation in the present work is the addition of real-time Semantic processing in contextual information. The benefit that characterizes the proposed approach is that an intelligent infrastructure for knowledge representation such as the Priamos

middleware can enable its (re)use in numerous ways. Among our most important observations is that the task of automated annotation in the Semantic Web community is almost similar to the challenge of contextualizing information in context-aware systems in a common and reusable way. The use of Semantic Web techniques in context aware systems can add intelligence to them thus enabling the development of innovative applications. The performance measurements demonstrate that inference-based systems have reached a mature state and can be used in order to provide intelligent results.

7 Future Work

Furthermore, the benefits that such a system provides are not limited to real-time processing. In many cases off-line searching on large bulks of multimedia and context data may be very time consuming unless treated at a semantic level. Everyday experience has shown that in emergency situations currently existing systems do not provide the mechanisms for fast, intelligent searching.

We are currently studying the integration with various types of sensors that are commonly available such as wireless sensor networks (Bluetooth, GPS, RFID), since the available infrastructure already incorporates the capability of handling incoming information contained in Web Service messages. Our first results towards this direction are presented in [57].

Another future goal includes the strengthening of the system's security level. Especially in surveillance scenarios or in scenarios where the transmitted information is sensitive, the system should present certain security features such as confidentiality, integrity or authentication. These features dictate the injection of security algorithms and techniques in every step of the information flow. Specifically, the way the information is transmitted, stored and accessed must be secured. As far as it regards transmission, the system could support encrypted XML messages, Secure Sockets Layer (SSL) or require user credibility in the Web Service layer. Access could be restricted with the use of identity certificates and finally, storage could benefit of the advances in cryptography as well.

Another improvement in the system's behaviour can be achieved by adjusting a sampling ratio or a temporary storage in the incoming messages. If for instance a sensor transmits messages in a rate higher than the one in which the middleware can process, sampling or buffering of the messages could fine-tune response times. The difference between these approaches is that sampling will reject a number of incoming messages while caching will store them for future process.

A more flexible rule engine implementation can also be of benefit. Rule language expressiveness can be enriched by adding more constructs in both the languages. We can also notice that in the current implementation the rule structure is flat for both the Mapping and the Semantic rules, in the sense that the rules are all applied to each incoming message. This poses no restrictions in creating a hierarchical structure in the Mapping rules, by adding extra conjunctive clauses in the beginning of each rule, in which case the rule engine will not parse the rest of the rule. However, for the Semantic rules, this will not be efficient since a simple `"if,class has individuals,..."` query can be very expensive. A Semantic rule hierarchy would allow for improved performance results compared to the ones in Figure 7(b).

Also, the future integration with offline (asynchronous) processing techniques on the *persistent* ontology model can increase the system's speed by improving mainte-

nance operations such as removing dated information. It would also be interesting to improve the current maintenance operations, in order for instance to treat probable inconsistencies that may appear due to erroneous tracker measurements.

As an open issue still remains the semantic query execution. The recent standardization[15] of SPARQL [53] and its adoption as the main Semantic Web query language allows for dynamic graphical interfaces that can build queries without imposing an additional knowledge overhead.

Moreover, the incorporation of Semantic Web Services by semantically describing the interfaces offered by the proposed system can lead to the development of applications that are trully semantically enriched through all steps of information processing. The Semantic Web Service Framework (SWSF) seems to be the most promising approach since it has been submitted to W3C[16] but no official recommendation has occurred yet.

Finally, it should be mentioned that systems based on the proposed middleware architecture have the flexibility to comprise various types of sensors. The use of the system presented in this paper presents cameras but the incoming information can flow in the middleware through sensors of temperature, humidity, A/V sensors, location sensors or even various environmental sensors such as oxygen, pollution and agricultural sensors. In other words, the system's architecture allows for distinct sensor types and architectures to be employed in the Data Acquisition Layer of Figure 1.

Appendix A: The Priamos Mapping Rule Language Syntax

Below is provided the grammar for the Priamos Rule language. We use the BNF metasyntax to clarify the use of both subsets of the rule language: the Mapping and the Semantic Rule Language.

```
<if> ::= ( <xml element exists>
         | <xml element has value>
         | <xml element has siblings> )
<xml element exists> ::= XPathExpression ( <and> | <then> )
<xml element has value> ::= XPathExpression
         ( gt | lt | gte | lte | eq | neq )
         ( Integer | String ) ( <and> | <then> )
<xml element has siblings> ::= XPathExpression
                               ( <and> | <then> )
<and> ::= ( <xml element exists>
          | <xml element has value>
          | <xml element has siblings> )

<then> ::= <insert individual in class>
         | <insert exactly one individual in class>
         | <execute SPARQL query>
<insert individual in class> ::= OntClass [ <named after> ]
                   [ <and set object property> |
                   ( <and set datatype property> )* ]
<named after> ::= XPathExpression
<insert exactly one individual in class> ::= OntClass
<and set object property> ::= OntProperty
```

[15] Note that due to the openness of the Web, no standards can be imposed or strictly followed. The closest approach to standardisation are the W3C or other consortia's *recommendations*.

[16] SWSF Overview: `http://www.w3.org/Submission/SWSF/`

```
                ( <last inserted individual in class> |
                <specific individual in class> )
<last inserted individual in class> ::= OntClass
<specific individual in class> ::= Individual
<and set datatype property> ::= OntProperty
                ( XPathExpression | String )
<execute SPARQL query> ::= String [ <and set ?variable value> ]
<and set ?variable value> ::= ( XPathExpression | String )
```

Appendix B: The Priamos Semantic Rule Language Syntax

```
<if> ::= ( <class has individuals>
         | <class has subclasses>
         | <number of individuals in class>
         | <datatype property in class>
         | <SPARQL query has results>
         | <SPARQL query does not have results> )
<class has individuals> ::= OntClass ( <and> | <then> )
<class has subclasses> ::= OntClass ( <and> | <then> )
<number of individuals in class> ::= OntClass
          ( gt | lt | gte | lte | eq | neq )
          (Integer | String) ( <and> | <then> )
<datatype property in class> ::= OntClass OntProperty
          ( gt | lt | gte | lte | eq | neq )
          ( Integer | String ) ( <and> | <then> )
<SPARQL query has results> ::= String ( <and> | <then> )
<and> ::= ( <class has individuals>
          | <class has subclasses>
          | <number of individuals in class>
          | <datatype property in class>
          | <SPARQL query has results>
          | <SPARQL query does not have results> )

<then> ::= ( <insert individual in class>
           | <insert exactly one individual in class>
           | <insert subclass of>
           | <execute a system command>
           | <execute a system command once>
           | <send Web Service message>
           | <execute SPARQL query> )
<insert individual in class> ::= OntClass
<insert exactly one individual in class> ::= OntClass
<insert subclass of> ::= OntClass String
<execute a system command> ::= PathToExecutable
<execute a system command once> ::= PathToExecutable
<send Web Service message> ::= String
<execute SPARQL query> ::=  String [ <and set ?variable value> ]
<and set ?variable value> ::= String
```

For both languages, `String` and `Integer` denote strings and integers respectively in programming language terms. `OntClass` and `OntProperty` refer to the full URIs or the QNames of classes and properties of the ontology, respectively. `PathToExecutable` is the full path to an executable file of the Operating System's filesystem, and finally, `XPathExpression` is an XPath expression used to select nodes from an XML document.

Acknowledgment

The work presented in this paper is carried out within the Priamos project, sponsored by the Greek General Secretarial of Research and Technology "Image, Sound, Language" research and development action. The authors of this paper would like to thank their colleagues in Athens Information Technology (`http://www.ait.edu.gr/`) for their collaboration.

References

1. Iria, J., Ciravegna, F., Cimiano, P., Lavelli, A., Motta, E., Gilardoni, L., Mönch, E.: Integrating Information Extraction, Ontology Learning and Semantic Browsing into Organizational Knowledge Processes. In: Proc. of the EKAW Workshop on the Application of Language and Semantic Technologies to support Knowledge Management Processes, at the 14th int'l. conf. on Knowledge Engineering and Knowledge Management (2004)
2. Uren, V., Cimiano, P., Iria, J., Handschuh, S., Vargas-Vera, M., Motta, E., Ciravegna, F.: Semantic Annotation for Knowledge Management: Requirements and a Survey of the State of the Art. Journal of Web Semantics, Elsevier **4**(1), 14–28 (2006)
3. Vembu, S., Kiesel, M., Sintek, M., Baumann, S.: Towards bridging the semantic gap in multimedia annotation and retrieval. In: Proc. of the 1st International Workshop on Semantic Web Annotations for Multimedia, SWAMM 2006 at the 15th int'l World Wide Web Conference, 2006 (2006)
4. Francois, A.R.J., Nevatia, R., Hobbs, J., , Bolles, R.C.: VERL: An ontology framework for representing and annotating video events. IEEE MultiMedia **12**(4), 76–86 (2005)
5. Dougherty, E., Laplante, P.: Introduction to Real-Time Imaging, chap. What is Real-Time Processing?, pp. 1–9. Wiley-IEEE Press (1995)
6. Dey, A.: Understanding and using context. Personal and Ubiquitous Computing **5**(1), 4–7 (2001). DOI http://dx.doi.org/10.1007/s007790170019
7. Lassila, O., Khushraj, D.: Contextualizing Applications via Semantic Middleware. In: Proc. of the The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MOBIQUITOUS'05), pp. 183–191. IEEE Computer Society, Washington, DC, USA (2005).
8. Dou, D., Pan, J., Qin, H., LePendu, P.: Towards Populating and Querying the Semantic Web. In: Proc. of 2nd International workshop on Scalable Semantic Web Knowledge Base Systems (SSWS'06), pp. 129–142 (2006). Co-located with ISWC 2006
9. Stamou, G., van Ossenbruggen, J., Pan, J., Schreiber, G., Smith, J.: Multimedia Annotations on the Semantic Web. Multimedia, IEEE **13**(1), 86–90 (2006)
10. Viola, P., Jones, M.: Rapid Object Detection using a Boosted Cascade of Simple Features. In: IEEE Conf. on Computer Vision and Pattern Recognition, pp. 511–518. Hawaii (2001)
11. Turk, M., Pentland, A.: Eigenfaces for recognition. Journal of cognitive neuroscience **3**(1), 71–86 (1991)
12. Belhumeur, P.N., Hespanha, J.P., Kriegman, D.J.: Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection. IEEE Transactions on pattern analysis and machine intelligence **19**(7), 711–720 (1997)
13. Bartlett, M.S., Movellan, J.R., Sejnowski, T.J.: Face Recognition by Independent Component Analysis. IEEE Transactions on neural networks **13**(6), 1450–1464 (2002)
14. Comaniciu, D., Meer, P.: Mean shift analysis and applications. In: Proc. of the Seventh IEEE int'l conf. on Computer Vision (ICCV'99), vol. 2, pp. 1197–1203. Kerkyra, Greece (1999)
15. Bradski, G.R.: Computer Vision Face Tracking for Use in a Perceptual User Interface. Intel Technology Journal (1998)
16. Allen, B.D., Bishop, G., Welch, G.: Tracking: Beyond 15 minutes of thought. SIGGRAPH Course Pack (2001)
17. Sohn, J., Kim, N.S., Sung, W.: A Statistical Model-based Voice Activity Detection. IEEE Signal Processing Letters **6**(1), 1–3 (1999)
18. Liu, S., Xu, M., Yi, H., Chia, L.T., Rajan, D.: Multimodal Semantic Analysis and Annotation for Basketball Video. EURASIP Journal on Applied Signal Processing **2006**, 1–13 (2006)

19. Lien, C.C., Chiang, C.L., Lee, C.H.: Scene-based Event Detection for Baseball Videos. Journal of Visual Communication and Image Representation **18**(1), 1–14 (2007)
20. Zhang, D., Chang, S.F.: Event Detection in Baseball Video Using Superimposed Caption Recognition. In: Proc. of the tenth ACM int'l conf. on Multimedia, pp. 315–318. ACM New York, NY, USA (2002)
21. Schroeter, R., Hunter, J., Guerin, J., Khan, I., Henderson, M.: A Synchronous Multimedia Annotation System for Secure Collaboratories. In: Proceedings of the Second IEEE int'l conf. on e-Science and Grid Computing (E-SCIENCE'06), p. 41. IEEE Computer Society, Washington, DC, USA (2006)
22. Petridis, K., Anastasopoulos, D., Saathoff, C., Timmermann, N., Kompatsiaris, I., Staab, S.: M-OntoMat-Annotizer: Image Annotation. Linking Ontologies and Multimedia Low-Level Features. In: Engineered Applications of Semantic Web Session (SWEA) at the 10th int'l conf. on Knowledge-Based Intelligent Information and Engineering Systems (KES'06). Bournemouth, U.K. (2006)
23. Chakravarthy, A., Ciravegna, F., Lanfranchi, V.: Cross-media document annotation and enrichment. In: Proc. 1st Semantic Web Authoring and Annotation Workshop (SAAW2006) (2006)
24. Etzioni, O., Cafarella, M., Downey, D., Popescu, A.M., Shaked, T., Soderland, S., Weld, D., Yates, A.: Unsupervised Named-Entity Extraction from the Web: An Experimental Study. Artificial Intelligence **165**(1), 91–134 (2005). DOI http://dx.doi.org/10.1016/j.artint.2005.03.001
25. Black, W., McNaught, J., Vasilakopoulos, A., Zervanou, K., Rinaldi, F.: CAFETIERE: Conceptual Annotations for Facts, Events, Terms, Individual Entities and RElations. Tech. rep., UMIST, Manchester, UK (2003). Parmenides TR-U4, 3.1
26. Buitelaar, P., Olejnik, D., Sintek, M.: A protégé plug-in for ontology extraction from text based on linguistic analysis. In: First European Semantic Web Symposium (ESWS). Heraklion, Greece (2004)
27. Goasdoué, F., Reynaud, C.: Modeling Information Sources for Information Integration. In: 11th European Workshop on Knowledge Acquisition, Modeling and Management (EKAW 99), *Lecture Notes in Artificial Intelligence*, vol. 1621, pp. 121–138. Springer-Verlag, Dagstuhl Castle, Germany (1999)
28. Kaykova, O., Khriyenko, O., Kovtun, D., Naumenko, A., Terziyan, V., A, A.Z.: General Adaption Framework: Enabling Interoperability for Industrial Web Resources. International Journal on Semantic Web and Information Systems **1**(3), 31–63 (2005)
29. Kagal, L., Finin, T., Johshi, A.: A Policy Language for Pervasive Computing Environment. In: Proc. of IEEE fourth International Workshop on Policies for Distributed Systems and Networks (POLICY'03) (2003)
30. Chen, H., Finin, T., Joshi, A.: Semantic web in the context broker architecture. In: PERCOM '04: Proc. of the Second IEEE Int'l Conf. on Pervasive Computing and Communications (PerCom'04), p. 277. IEEE Computer Society, Washington, DC, USA (2004)
31. Dey, A., Abowd, G., Salber, D.: A context-based infrastructure for smart environments. In: First Int'l Workshop on Managing Interactions in Smart Environments (MANSE 99), pp. 114–128. Dublin, Ireland (1999)
32. Pandis, I., Soldatos, J., Paar, A., Reuter, J., Carras, M., Polymenakos, L.: An Ontology-based Framework for Dynamic Resource Management in Ubiquitous Computing Environments. In: Proc. of the 2nd int'l conf. on Embedded Software and Systems (ICESS'05) (2005)
33. Masuoka, R., Parsia, B., Labrou, Y.: Task Computing – The Semantic Web Meets Pervasive Computing. In: 2nd int'l Semantic Web Conference (ISWC'03). Sanibel Island, Florida, USA (2003)
34. Román, M., Hess, C., Cerqueira, R., Ranganathan, A., Campbell, R., Nahrstedt, K.: Gaia: A Middleware Platform for Active Spaces. ACM SIGMOBILE Mobile Computing and Communications Review **6**(4), 65–67 (2002)
35. Vazquez, J.I., de Ipiña, D.L., nigo Sedano, I.: Computational Science and Its Applications – ICCSA 2006, Workshop on Ubiquitous Web Systems and Intelligence (UWSI 2006), *Lecture Notes in Computer Science*, vol. 3983/2006, chap. SOAM: An Environment Adaptation Model for the Pervasive Semantic Web, pp. 108–117. Springer Berlin / Heidelberg (2006)
36. Vu, V.T., Bremond, F., Thonnat, M.: Automatic video interpretation: A novel algorithm for temporal scenario recognition. In: Proc. of the 18th int'l Joint conf. on Artificial Intelligence (IJCAI 2003), pp. 1295–1302

37. Konstantinou, N., Solidakis, E., Zoi, S., Zafeiropoulos, A., Stathopoulos, P., Mitrou, N.: Priamos: A Middleware Architecture for Real-Time Semantic Annotation of Context Features. In: IET int'l conf. on Intelligent Environments (IE'07), pp. 96–103. Ulm, Germany (2007)
38. Cardoso, J.: The Semantic Web Vision: Where are We? IEEE Intelligent Systems, **22**(5), 84–88 (2007)
39. Carroll, J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., Wilkinson, K.: Jena: Implementing the Semantic Web Recommendations. Tech. Rep. HPL-2003-146, Hewlett-Packard (2003)
40. Patel-Schneider, P., Horrocks, I.: OWL Web Ontology Language: Semantics and Abstract Syntax Section 3. Direct Model-Theoretic Semantics. Available online at (2004). `http://www.w3.org/TR/owl-semantics/direct.html#3.1`
41. Papamarkos, G., Poulovassilis, A., Wood, P.T.: Event-Condition-Action Rule Languages for the Semantic Web. In: Workshop on Semantic Web and Databases (SWDB 03), pp. 309–327 (2003)
42. May, W., Alferes, J., Amador, R.: Rules and Rule Markup Languages for the Semantic Web, *Lecture Notes in Computer Science*, vol. 3791, chap. Active Rules in the Semantic Web: Dealing with Language Heterogeneity, pp. 30–44. Springer Berlin / Heidelberg (2005)
43. Hirtle, D., Boley, H., Grosof, B., Kifer, M., Sintek, M., Tabet, S., Wagner, G.: Schema Specification of RuleML 0.91. available online at `http://www.ruleml.org/0.91/` (2006)
44. Baader, F., Nutt, W.: The Description Logic Handbook, chap. Basic Description Logics, pp. 47–100. Cambridge University Press (2002)
45. Motik, B., Sattler, U.: A Comparison of Reasoning Techniques for Querying Large Description Logic ABoxes. In: M. Hermann, A. Voronkov (eds.) Proc. of the 13th int'l. conf. on Logic for Programming Artificial Intelligence and Reasoning (LPAR'06), *LNCS*, vol. 4246, pp. 227–241. Springer, Phnom Penh, Cambodia (2006)
46. Sirin, E., Parsia, B., Grau, B., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Journal of Web Semantics: Science, Services and Agents on the World Wide Web **5**(2), 51–53 (2007)
47. Horrocks, I.: Using an Expressive Description Logic: FaCT or Fiction? In: A.G. Cohn, L. Schubert, S. Shapiro (eds.) Principles of Knowledge Representation and Reasoning: Proceedings of the 6th Int'l Conference (KR-98), pp. 636–647. Morgan Kaufman (1998)
48. Bechhofer, S.: DIG 2.0: The DIG Description Logic Interface. Available online at `http://dig.cs.manchester.ac.uk/` (2006)
49. Beckett, D.: RDF/XML Syntax Specification(Revised). Available online at `http://www.w3.org/TR/rdf-syntax-grammar/` (2004). W3C Recommendation
50. Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D., Patel-Schneider, P., Stein, L.A.: OWL Web Ontology Language Reference. World Wide Web Consortium, Recommendation REC-owl-ref-20040210 (2004)
51. Horrocks, I., Patel-Schneider, P., van Harmelen, F.: From $\mathcal{SHIQ}$ and RDF to OWL: The Making of a Web Ontology Language. Journal of Web Semantics: Science, Services and Agents on the World Wide Web **1**(1), 7–26 (2003)
52. Toninelli, A., Montanari, R., Kagal, L., Lassila, O.: The Semantic Web - ISWC 2006, *Lecture Notes in Computer Science*, vol. 4273, chap. A Semantic Context-Aware Access Control Framework for Secure Collaborations in Pervasive Computing Environments, pp. 473–486. Springer Berlin / Heidelberg (2006)
53. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF (2008). `http://www.w3.org/TR/rdf-sparql-query/`
54. Seaborne, A., Manjunath, G.: SPARQL/Update: A language for updating RDF graphs (2008). `http://jena.hpl.hp.com/~afs/SPARQL-Update.html`
55. Stergiou, A., Pnevmatikakis, A., Polymenakos, L.: The AIT Multimodal Person Identification System for CLEAR 2007, In: Multimodal Technologies for Perception of Humans, pp. 221–232 (2007)
56. Karame, G., Stergiou, A., Katsarakis, N., Papageorgiou, P., Pnevmatikakis, A.: 2D and 3D Face Localization for Complex Scenes. In: Advanced Video and Signal Based Surveillance, 2007. AVSS 2007. IEEE Conference on, pp. 371–376 (2007)
57. Zafeiropoulos, A., Konstantinou, N., Arkoulis, S., Spanos, D.E., Mitrou, N.: A Semantic-based Architecture for Sensor Data Fusion. In: 2nd int'l conf. on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM'08), pp. 116–121. Valencia, Spain (2008)