PRIAMOS: A MIDDLEWARE ARCHITECTURE FOR REAL-TIME SEMANTIC ANNOTATION OF CONTEXT FEATURES

Nikolaos Konstantinou, Emmanuel Solidakis, Stavroula Zoi, Anastasios Zafeiropoulos, Panagiotis Stathopoulos, Nikolas Mitrou

National Technical University of Athens, Heroon Polytechneiou Str, 15773 Zografou, Athens, Greece
Email:
{nkons;tzafeir}@cn.ntua.gr, {esolid;vzoi;pstath}@telecom.ntua.gr, mitrou@softlab.ntua.gr

Keywords: Real-time, automatic, rule-based, annotation, semantics, context features, middleware

Abstract

This paper proposes a middleware architecture for the automated, real-time, unsupervised annotation of low-level context features and their mapping to high-level semantics. The distinguishing characteristic of this 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 are pluggable to the middleware architecture thus facilitating application development and system configuration to different real-world scenarios. A prototype implementation based on Semantic Web tools is presented in depth, while the benefits and drawbacks of this approach are underlined. We argue that the use of Semantic Web provides powerful answers to context awareness challenges. Furthermore, it enables the composition of simple rules through human-centric interfaces, which may launch a context-aware system that will annotate content without the need for user technical expertise.
A test case of system operation in a laboratory environment is presented. Emphasis is given, along with the theoretical justification, to practical issues that arise in real-world scenarios.

1 Introduction

The basic concept of the Semantic Web contribution is annotation of content in order to become easily retrievable. This purpose is served by a number of prevalent Semantic Web technologies like content description languages, query languages and annotation frameworks.
Web content of different types is usually annotated with the use of keywords. Famous corporations follow this approach such as flickr.com for pictures, del.icio.us for user bookmarks and youtube.com for video content. Here it must be noted that annotation is kept separately from the data, an approach that is also compliant with the Semantic Web model.
Despite its importance, annotation is not always present due to several factors. First, it is a time-consuming task and users do not usually consider it important enough to spend time annotating the already published content. The companies on the other hand mostly 'believe' that annotation is a loss of resources in terms of time and money. Moreover, the reuse of this information is troublesome as annotation is usually likely to be redundant, partial or stored in different formats [19]. If we add to the above that annotation easily becomes out-of-date then we can easily state that the commercial future of the Semantic Web is endangered [10].
The automation of the whole annotation procedure will be a step further to its wider deployment. What we present in this paper is Priamos, a rule-based middleware system for real-time annotation of context features, implemented following Semantic Web standards and cutting-edge web technologies. The distinguishing characteristic of this 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 are pluggable to the middleware architecture thus facilitating application development and system configuration to different real-world scenarios. Furthermore, we demonstrate how we can setup simple rules through human-centric interfaces in order to launch a context-aware system that will annotate content without the need for user technical expertise.
A domain that is not so obviously related to semantic web is the domain of context-aware systems. We analyze in section 3 the need in these systems for a high-level description of the context, of the world according to their perception. We will see that the semantic description of the contextual information is the best choice for a number of reasons. Nevertheless, the approach presented in this paper is twofold because on the one hand we present a solution for automated annotation based on Semantic Web technologies, but on the other hand we combine the approach with context awareness.
In section 2 we present the latest advances in the field of context annotation and compare our approach to existing works in the area of context-aware systems. Section 3 depicts the thinking motive that led to the design and implementation of the Priamos middleware in its current form. We continue in chapter 4 analyzing the overall

abstract architecture and the software implementation. In section 5 we demonstrate a test case of the system, operating in a laboratory environment. Finally, we conclude in section 6 by noticing the future directions of expanding the presented work.

2 Related work and motivation

Generally speaking, there are two kinds of approaches regarding content – multimedia or not – annotation. The proposed approaches can be divided in manual or automated according to whether the annotation requires human intervention, usually aided by semi-automatic information extraction algorithms, or the procedure is fully automated.

The first approach is manual annotation, usually aided by semiautomatic metadata extraction techniques. This category contains tools like Vannotea [11] that annotates collections of images, video, audio or 3D objects and M-Ontomat Annotizer [13] that is part of the CREAM framework [1] and one of the major outcomes of the IST project aceMedia. M-Ontomat can link low-level MPEG-7 visual descriptors with RDF(S) ontologies and offers the possibility of annotating the Deep Web. COHSE [14] is an ongoing work that aims at annotating content at retrieval time, as readers browse the documents, or at authoring time, as readers author the documents. Amaya, W3C's annotation-friendly editor/browser and SMORE [15] for Web content, also fall to this category.

Automatic annotation systems can be further divided into two categories: user-centered and pattern- or rule-centered. User-centered can be divided into supervised and unsupervised. Users of automatic annotation systems need to be aware of their limitations, like missing annotations (known technically as low recall) and incorrect annotations (known as low precision), and the trade off against each other.

Supervised approaches include MnM [20] that provides an environment to manually annotate data, although its initial design aimed at marking up training data for Information Extraction tools. Melita [26] is a user-driven automated semantic annotation tool which is supported by Amilcare, an information extraction engine. The aim of the project is to gradually change the role of the user in the annotation process.

Unsupervised approaches include Armadillo [19] and KnowItAll [21] that automate information extraction in a similar way. In the SmartWeb project they are investigating an unsupervised approach for RDF knowledge base population[1].

As far as it concerns pattern-based and rule-based approaches, except from Priamos, we can see only a few approaches in the bibliography. These include CAFETIERE [25], which is a rule-based system for generating XML annotations and was developed as part of the Parmenides project and does not make use of Semantic Web technologies.

From the context-aware point of view, related to the Priamos concept are systems that use ontological descriptions to express contextual information. Older approaches that investigated various aspects of the context-aware computing, like Ponder, the Context Toolkit, HP's CoolTown and the Intelligent Room project did not use a formal model to represent context information. Interest in a formal common way of representing context information has been shown lately.

The IST project CHIL[2] is based on multimodal perceptual user interfaces and aims at supporting human-to-human interaction. In CHIL the description of the world model is based on the core vocabulary of the CHIL OWL ontology for controlling sensors and actuators [24].

The KaOS project [30] uses Description Logics ontologies as the basis for representing and reasoning about policies. Nevertheless, application-specific ontologies must be built on top of the existing ones. The also well-known Rei framework [31] uses RDF(S) or OWL Lite to represent context but the specification is limited to the terms of the Rei Ontology.

From the scope of the pervasive and ubiquitous systems, we could compare Priamos with CoBrA [33]. Semantic Web technologies specialized for ubiquitous computing have also been applied in several environments such as Masuoka (Task Computing) [34], Gaia [35] and the SoaM Architecture [36].

3 Using Semantic Web Technologies to Support Context Awareness

Context means situational information. According to [18], "Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves." A system is context-aware if it can extract, interpret and use context information and adapt its functionality to the current context of use.

The challenge for context-aware systems lies in the complexity of capturing, representing and processing contextual data, such as location, ongoing activities etc. captured by sensors and appropriate software. A common representation format should be adopted, in order for different applications to be able to use the same context information. Thus, to ensure syntactic and semantic interoperability, we adopt a Semantic Web compliant approach, that allows us the future aggregation with various heterogeneous data sources.

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 this approach is that a world model can be bound to a reasoner and deduce implicit knowledge, adding intelligence to the system. Moreover, we have to notice that context-aware systems are often complicated enough to the point that tasks like annotation and decision making, become

[1] Home of the SmartWeb project: http://www.smartweb-project.de/, last accessed on 25-03-07

[2] Home of the CHIL project: http://chil.server.de/, last accessed on 25-03-07

unmanageable if not supported by automated procedures. Finally, it is undisputed that the use of middleware facilitates context representation and processing at the infrastructure level, better enabling reuse of derived context by multiple data consumers. Ad hoc formalisms with insufficiently established semantics make context aggregation difficult [22].
Consecutively, one of the most challenging issues of context aware applications is the inclusion of intelligence while processing the incoming information and deducting meaning. Below, we analyze the most interesting elements among Semantic Web technologies that can potentially play a crucial role in context-aware applications.

3.1 Rule languages

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 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.
RuleML is an implementation of rules used for deduction, rewriting, and further inferential-transformational tasks. In general, it is a specification for immediate rule interchange and can be gradually extended, possibly together with related initiatives, towards a proposal that could be submitted to the W3C[3]. Rules in RuleML are stated in a combination of natural language and formal notation. Further purposes of rule markup include the providing of a rule interchange format for exchanging rules between different tools/systems, the marking up of rule content in business documents and the providing of a high-level specification language for active content in the Web. Though, very few implementations, like Mandarax and OO jDrew are available.
SWRL [6], the Semantic Web Rule Language is based on a combination of the OWL DL and OWL Lite sublanguages of OWL with the Unary/Binary Datalog RuleML sublanguages of the Rule Markup Language. The proposal extends the set of OWL axioms to include Horn-like rules. SWRL has been implemented as a Protégé plugin and is shipped with the full version under the name SWRLTab [27], it has been investigated by projects like SweetRules[4] and it is supported by KAON2 [4], Pellet [5] and RacerPro reasoners. It has been submitted to W3C for standardization, no standard has been adopted yet, though, and the support by reasoners is very limited at the time.
The most significant problem is the freedom in the offered expressivity by rules that frequently makes it undecidable. In other words, the subset of a rule language that can be used to reassure finite reasoning procedures is restricted. The makers of Pellet proposed a subset of safe SWRL rules [9][29] that assure decidability but the support of SWRL or RuleML in applications is still an ongoing procedure.

3.2 The time dimension

Temporal ontologies expand the current ontologies adding the time dimension. The dimension of time exists in almost every real-world application. Specifically in the web services, a timestamp of every transaction is essential. Having this in mind, the Semantic Web Best Practices and Deployment Working Group3 developed the DAML-Time ontology that today evolved to OWL-Time [17].
OWL-Time is an ontology that provides a vocabulary for expressing facts about topological relations among instants and intervals, together with information about durations, and about date-time information. The concepts described in this ontology can describe time in web services, hypertext documents or even custom applications. In general, the Time ontology contains concepts like `Instant` for time instances and `Interval` for time intervals. These concepts have properties like `starts`, `ends`, `intOverlaps`, `TimeZone`, `unitType` and it is an approach that should satisfy most of applications' needs.
T-Owl[5] is a starting project promising to analyze the stock market in real-time with the use of temporal ontologies. A time ontology is presented in [2]. It is possible to use such an ontology for application-specific purposes just by embedding it to the application ontology. The OpenCyc ontology [32] also makes special reference to time, by including concepts like `TemporalThing` and `Event`. Similarly, time concepts are also supported in the SOUPA ontology [37].
We underline that the form of the time dimension in Description Logics, i.e. OWL ontologies, can be added to the ontology with a simple import of the time ontology. Thus, such an addition that is a critical issue in context awareness can be achieved with no extra effort with the use of Semantic Web technologies.

4 The Priamos middleware architecture

The Priamos middleware architecture comprises a set of core reusable distributed components for the automated, real-time annotation of low-level context features and their mapping to high-level semantics. The main idea is to launch a procedure that annotates contextual information upon its appearance. The resulting Knowledge Base will reflect a spherical perception of the world model.
First of all, the Priamos architecture abstracts the outputs from heterogeneous, low-level 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 and is automatically initiated without any further human intervention.

[3] The Rule Markup Initiative: http://www.ruleml.org/
[4] The SweetRules project along with an extent variety of open source projects for the Semantic Web is hosted at http://projects.semwebcentral.org/
[5] Home of the TOWL Project: http://www.towl.org

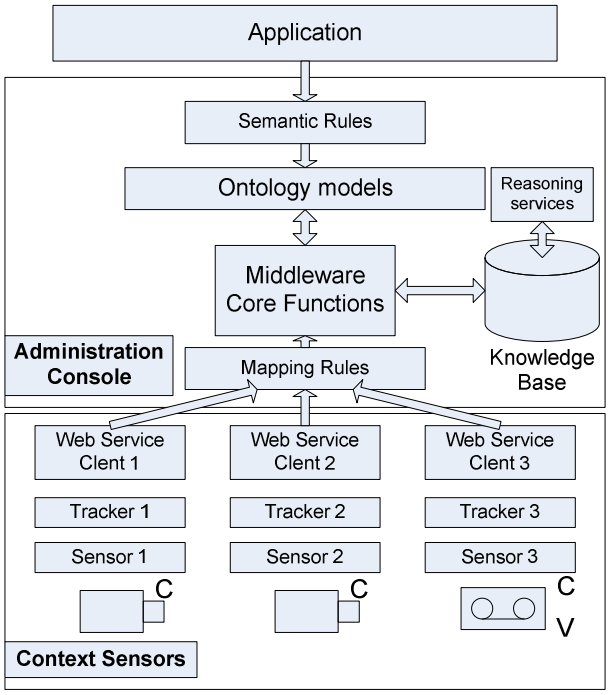

Fig. 2: The Priamos middleware architecture

The basic components of the system are the data sources (e.g. sensors) combined with low-level feature extraction components (trackers), the user terminals running the applications, the administration console, and the server (Fig. 2). The Priamos middleware components are distributed to all these components, as illustrated in Fig. 2. 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 Priamos server and a Knowledge Base is created where all the features of interest captured by the sensors are gathered.

The architecture does not constrain the system to be used only for multimedia content. It can as well be used for all cases where context awareness is crucial, for example to monitor user clicks in a web environment.

During the design of Priamos we faced the dilemma of whether to implement a multi-agent system or a distributed architecture based on web services. Although a multi-agent system would be as much as flexible and scalable as the current web services-based implementation [28], we chose not to restrict further development to the limitations of a specific agent framework.

4.1 Software modules

The Priamos modular software architecture ensures its extendibility and adaptation to newer technologies. It mainly comprises: an exported web service, the message templates, the ontology models, a set of mapping rules, a set of semantic rules, the external reasoning server and finally, the trackers.

Web Service. The Web service module is responsible for message manipulation. The only requirement is that messages are expressed in any arbitrary well-formed XML document. When a message arrives, it is processed by the mapping rules that the user has entered to describe the desirable behaviour. In their current form, mapping rules state simple rules of the form: “if an xml element exists then insert an individual in one of the ontology classes”. In this way we can declare a mapping rule that inserts an individual in a class e.g. `Persons` every time we receive an xml containing the element message e.g. `Human`.

Message Templates. The received messages can conform to any chosen specifications. As previously mentioned, the only necessity is that the messages are well formed XML documents. Message information concerns environment elements such as person locations or sound volume.

Ontology models. The database model is stored using Jena[3] 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 triples that, in semantic web terms, are called statements and form the underlying graph of the model.

The annotation is kept in a Knowledge Base, separately from the incoming data that could be of any form (e.g. simple text or multimedia). Links to them are stored making possible a future retrieval.

XML Mapping Rules. The XML Mapping rules fetch data from the XML message and store it into the ontology model in the form of class individuals. The rules rely on the fact that for every XML element there is a unique XPath expression that retrieves its value. The rules are formed according to the following pattern:

```
if condition then action
```

That is very similar to a Horn clause but, in this case, the predicates in the head are represented by classes of the ontology and predicates in the body by nodes of the XML tree. The approach is similar to the one presented in [23]. In Description Logic terms we could state that the mapping rules define how the ABox of the Knowledge Base is populated.

Semantic Rules. The rule-based approach is adopted because of its extendable and adaptive nature. The current implementation consists of custom rules for the reasons discussed in §3.1. The rules here also abide by the above mentioned condition/action pattern but predicates now are represented by classes, properties and individuals defined in the application-specific ontologies. We are currently working in extending the current implementation keeping in mind that the expressivity supported by rules can easily lead to non terminable loops.

The Reasoning Server.. There is a variety of available reasoners, commercial like RacerPro or OntoBroker, free of charge like KAON2 [4] and open-source like Pellet [5] and FaCT++. All of them support DIG [12] 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 Priamos, leaving the reasoner choice up to the user.

Open source reasoners provide the ability of integration in application code so that developers can embed in their applications Pellet or FaCT++ gaining a lot in speed. When the reasoner is integrated to the application there is no overhead caused by the necessary communication

through HTTP messages between the reasoner and the application. This approach, though, would restrict Priamos to a specific reasoner and would diminish our choices. In our implementation, we used Pellet, a sound and complete OWL DL reasoner, implemented in Java and based on optimized tableaux procedures.

4.2 Application description

The tools that are used to handle the various components are developed in Web environment and they consist of: 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 user decides which description language he should use: RDF(S) [16] or OWL [7]. Priamos ontologies have no 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[6] and it is usually more convenient to customize an ontology to an application's needs than to start authoring from scratch.

Message template manager. The messages that can be received are stored locally 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. The human-centric approach that we have followed led us to an implementation adopting the AJAX[7] methodology.

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 administrator can assign mapping rules to specific models. These rules will be processed one by one upon the arrival of message and they are responsible for adding the extra information in the ontology. The mapping interface displays the ontology hierarchy on the left, the XML tree on the right and the defining rules underneath.

An example of a mapping rule can order the system to check whether a specific element exists in an incoming message. 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 exists(Message/Tracker/Event/Person) then insertIndividualIn(foaf:Person)`", will insert an individual in the class `foaf:Person` if the path `Message/Event/Person` exists in the message.

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. An example of a rule that can be declared is "`if hasIndividuals(DangerousEvent) then set_alarm`". In this case, the system will call a predefined action named alarm (i.e. an external command) if the check for individuals in the class `DangerousEvent` returns true. A graphical authoring tool can be easily provided based on this rational for the composition of rules by non-expert users.

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 recognition, 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 to the main Priamos server.

4.3 Users and roles in Priamos

Priamos reusable core functions facilitate application development, in different scenarios and context configurations. The mapping of low-level features to high-level semantics enables the definition of different user roles, according to their experience and technical background. Since the users are not always domain or technical experts, it is important to be able to configure the system through human-centric interfaces. Knowledge overhead has been an important problem in Semantic Web applications in the past, making them unsuitable for non-expert users i.e. ordinary end users who are not necessarily familiar with domain-specific semantic data or ontologies.

Users that benefit from Priamos technology are classified into four categories: system administrators, middleware maintainers, application developers and end users.

System administrators. They have the overall supervision of the system's functions and can configure it for different operation scenarios. A system administrator can define features of interest to be captured (e.g. when a security alert should be triggered) through a high-level interface.

Middleware maintainers. A maintainer of the system is the domain expert, burdened with the task of defining the mapping rules from the incoming messages to the ontology concepts.

Application developers. They are the mostly benefited users by the use of the Priamos functions. Instead of developing application specific code each time, the application 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 and the responsibility to tune the system's behaviour as wished, through Priamos provided event handlers and callback functions.

End users. They can be users who are not familiar with technology at all. They can be simply monitoring a system operation session (e.g. a guardian in a security-

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

[7] AJAX, a shorthand for Asynchronous Javascript and XML, is a development technique for the creation of web applications of increased interactivity, speed and usability. AJAX can be seen in action in almost every Google application, such as Gmail, Maps, etc.

surveillance scenario, or waiting to receive automated notifications in form of a sound, an email, a call, an alert in general (e.g. a mobile user who receives alerts in his device).

4.4 Message processing cycle

When a message is received, it is first checked for its XML validity. The middleware poses no extra constraints. First, the message will be processed by the mapping rules. All of the mapping rules will be applied to the incoming message. We note that this procedure can possibly modify more than one of the models that lie in the database. As a result of the application of the mapping rules, the Knowledge Base is updated with the new facts.
Consecutively, the semantic rules are applied. The set of the semantic rules that correspond to the modified ontologies is now applied to them. As we noted before, this set of rules checks the conditions and performs actions related to the database models, no matter what were the contents of the XML message. This level of abstraction was chosen for two reasons, first because it separates XML mapping from semantic rules, facilitating the authoring process and second, because this processing phase can take advantage of the evolution of Semantic Web rule languages. As depicted in Fig. 3, the data is first aggregated and adapted and then it is consumed.

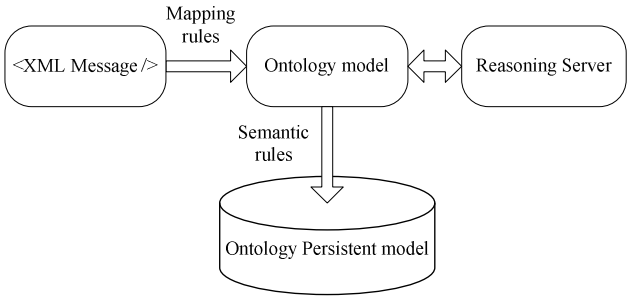

Fig. 3: The set of the mapping rules is first applied to each incoming message. Then, the set of the semantic rules are applied. As a result, the ontology model is expanded and the new facts are added to it.

After the message process has terminated, the persistent 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 will potentially be stored at the database after each message.

5 Test-case scenario

Based on the proposed architecture, we describe below a use-case scenario of the Priamos middleware when used to monitor a room and request an alert in case something "unusual" happens. The environment in our example consists of a series of cameras and microphones, the Priamos middleware and the Surveillance application[8]. The application developer can configure the system according to his consideration.
Phase 1: System Bootstrap. The system is activated and detects automatically the type of devices that are connected to it (cameras and microphones) as well as their topology. At the same time this environment is connected to the appropriate trackers. This process can be executed periodically or according to the requested needs e.g. when a new sensor is activated.
Phase 2: The middleware maintainer connects online to the control panel of the middleware which returns the trackers that it recognizes, e.g. a tracker for movement recognition and a tracker for speech-recognition. The middleware will return the following possible description for the end-user, in XML format, where the tag properties are prefixed with the @ character:
Body tracker that recognizes a human body in his visual range and returns his coordinates:

```
Message
  Tracker  @type
    TimeStamp  @value
    DataSource  @id  @name  @url
    person  @certainty  @id
      location  @datasourceId  @x  @y
```

In addition, the maintainer has control over the ontologies of the system. He may have ontologies describing divergent domains such as activities, security, time, persons etc. He could as well have one, describing a variety of concepts like the infamous OpenCyc [32], although this would slow things enough.
The maintainer will configure the incoming messages to correspond to ontology concepts. He will assign mapping rules in the condition/action pattern that we analyzed in §4.2. Let the example rule: `if Message/Tracker/TimeStamp/@value > "21.00" then insertIndividualIn(UnusualEvent)`, where `UnusualEvent` is a hypothetic class in the ontology model. As a result, after the definition of the set of rules, the system's interoperability with the outer environment is set. It is now the turn of the application developer to further configure the system.
Phase 3: The application developer confronts an automatically updating ontology. What he has to set is a set of rules to manage this growth. The constructors he uses to state the rules in the ontology contain only individuals, classes and properties. He can declare a set of rules, for example in the form: `if hasIndividuals(UnusualEvent) then contact_me`, where `contact_me` is an action that executes a shell command, e.g. `sendSMS content="Something strange is happening"`.
After the system has been configured, it is assured that the end user will receive a notification in case the system senses an event of interest. This approach has the

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

advantage that all users had to spend the minimum effort to configure the middleware to their needs. The actions that were taken were: to turn on the system, to map the incoming messages and to declare the required behaviour in terms of rules, all through the AJAX-based user-friendly web interface.

6 Conclusions and Future work

Among our most important observations is that it seems that two apparently different domains of active research share in fact many common aspects. The task of automated annotation in the Semantic Web community is almost similar to the challenge of contextualizing information in a common and reusable way in context-aware systems. It seems that both domains will benefit from the evolution of Semantic Web technologies. A common framework for knowledge representation will enable its (re)use in numerous ways.

The ongoing work presented in this paper aims at promoting research in the field of semantic annotation and context-aware systems. We defined the scope of our goals, discussed about current approaches and emerging problems and demonstrated an innovative middleware architecture based on Semantic Web standards.

In the future our goals include the exploitation of semantically described web services. A description of the Web Services in OWL-S will offer us more and better-described actions. Moreover, we will enhance the semantic rules that apply to the ontology, by updating the description vocabulary with more complex constructors and phrases. The extension in the same way will take place for the mapping rules, as well. It is also our intention to create a mechanism for offline semantic search of the stored data. For this purpose we will pay attention to the daily maintenance of the database data. We are also currently studying the effect of fuzziness on the events processed by the system and one of the future extensions will also be the probabilistic processing of information.

We also take seriously into consideration the extensibility of our system. We are challenged to provide answers to questions such as “What happens in case we should attend more than one parallel sessions in different conferences? How can we achieve the surveillance of places that are geographically remote among each other? We are considering the option of using local agents that will be responsible for the parallel action of several installed middlewares. In addition, we are working on real-world surveillance and smart-room scenarios. Finally, we are concerned about obtaining our first scalability results. Priamos will soon be subject of benchmarking measurements concerning latency in message processing relevantly to the number and the sizes of the ontologies, the messages’ rate of arrival and the number and complexity of mapping and semantic rules.

Acknowledgements

The work presented in this paper is carried out within the Priamos project, funded by the Hellenic GSRT. The authors of this paper would like to thank their colleagues in A.I.T. (http://www.ait.edu.gr/) for their collaboration.

References

[1] S. Handschuh, S. Staab, R. Studer: Leveraging metadata creation for the semantic web with Cream. In *Proceedings of the Annual German Conference on AI*, Berlin, Germany, 2003.

[2] V. Milea, F. Frasincar, U. Kaymak, T. di Noia: An OWL-Based Approach Towards Representing Time, *International Workshop on Web Information Systems Modeling*, Trondheim, Norway, 2007

[3] J.J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, K. Wilkinson: Jena: Implementing the Semantic Web recommendations. *Technical Report HPL- 2003-146*, Hewlett-Packard, 2003

[4] B. Motik, U. Sattler: A comparison of reasoning techniques for querying large description logic ABoxes. *Proceedings of the 13th International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR 2006)*, Phnom Penh, Cambodia, 2006

[5] E. Sirin, B. Parsia, B.C. Grau, A. Kalyanpur, Y. Katz: Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, 2006

[6] I. Horrocks, P.F. Patel-Schneider, H. Boley, S.T. Benjamin, G.M. Dean: SWRL: A Semantic Web rule language combining OWL and RuleML. *World Wide Web Consortium, Member Submission*, 2004

[7] S. Bechhofer, F. van Harmelen, J.A. Hendler, I. Horrocks, D.L. McGuinness, P.F. Patel-Schneider, L.A. Stein: OWL web ontology language reference. *World Wide Web Consortium, Recommendation* REC-owl-ref-20040210, 2004

[8] J.R. Hobbs, F. Pan: Time ontology in OWl. Available at http://www.w3.org/TR/owl-time/, 2006

[9] B. Parsia, E. Sirin, B.C. Grau, E. Ruckhaus, D. Hewlett: Cautiously approaching SWRL. Available at http://www.mindswap.org/papers/CautiousSWRL.pdf , 2005

[10] V. Uren, P. Cimiano, J. Iria, S. Handschuh, M. Vargas-Vera, E. Motta, F. Ciravegna: Semantic Annotation for Knowledge Management: Requirements and a Survey of the State of the Art *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 4, No. 1, pp. 14-28, 2006

[11] R. Schroeter, J. Hunter, J. Guerin, I. Khan and M. Henderson. "A Synchronous Multimedia Annotation System for Secure Collaboratories", *2nd IEEE International Conference on E-Science and Grid Computing (eScience’06)*, Amsterdam, Netherlands, 2006

[12] S. Bechhofer: DIG 2.0: The DIG Description Logic Interface, DIG Working Group Note, http://dig.cs.manchester.ac.uk/, 2006

[13] K. Petridis, D. Anastasopoulos, C. Saathoff, N. Timmermann, I. Kompatsiaris and S. Staab: M-OntoMat-Annotizer: Image Annotation. Linking Ontologies and Multimedia Low-Level Features,

Engineered Applications of Semantic Web Session (SWEA) at the 10th International Conference on Knowledge-Based & Intelligent Information & Engineering Systems (KES 2006), Bournemouth, U.K., 2006

[14] Y. Yesilada, S. Bechhofer, B. Horan: Personalised Dynamic Links on the Web, *1st International Workshop on Semantic Media Adaptation and Personalization (SAMP)*, 2006.

[15] A. Kalyanpur, B. Parsia, J. Hendler, J. Golbeck: SMORE - Semantic Markup, Ontology, and RDF Editor. Available online at http://www.mindswap.org/papers/SMORE.pdf, 2005

[16] D. Beckett, RDF/XML Syntax Specification(Revised) http://www.w3.org/TR/rdf-syntax-grammar/, 2004

[17] J.R. Hobbs, F. Pan: Time Ontology in OWL http://www.w3.org/TR/owl-time/, 2006

[18] A. Dey: Understanding and Using Context, *Journal of ubiquitous computing*, vol. 5, No. 1, pp. 4-7, 2001

[19] J. Iria, F. Ciravegna, Ph. Cimiano, A. Lavelli, E. Motta, L. Gilardoni, E. Mönch: Integrating Information Extraction, Ontology Learning and Semantic Browsing into Organizational Knowledge Processes. *Proceedings of the EKAW Workshop on the Application of Language and Semantic Technologies to support Knowledge Management Processes, at the 14th International Conference on Knowledge Engineering and Knowledge Management*, Whittlebury Hall, Northamptonshire, U.K., October 2004

[20] Vargas-Vera M., E. Motta, J. Domingue, M. Lanzoni, A. Stutt, F. Ciravegna, MnM: A tool for automatic support on semantic markup, *KMi Technical Report No. 133*, September 2003

[21] O. Etzioni, M.J. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D.S. Weld, A. Yates, Unsupervised named-entity extraction from the Web: an experimental study, *Journal of Artificial Intelligence*, vo1. 65, No. 1, pp. 91–134, 2005

[22] O. Lassila, D. Khusraj: Contextualizing applications via semantic middleware. *Proceedings of the 2nd annual conference on Mobile and Ubiquitous Systems: Networking and Services, (MobiQuitous'05)*, Washington D.C., USA, 2005

[23] A. Toninelli, R. Montanari, L. Kagal, O. Lassila: A Semantic Context-Aware Access Control Framework for Secure Collaborations in Pervasive Computing Environments. *International Semantic Web Conference (ISWC '06)*, Athens, Georgia, 2006

[24] I. Pandis, J. Soldatos, A. Paar, J. Reuter, M. Carras, L. Polymenakos: An ontology-based framework for dynamic resource management in ubiquitous computing environments. *2nd International Conference on Embedded Software and Systems (ICESS'05)*, 2005

[25] W.J. Black, J. McNaught, A. Vasilakopoulos, K. Zervanou, B. Theodoulidis and F. Rinaldi: Parmenides TR-U4,.3.1: CAFETIERE: Conceptual Annotations for Facts, Events, Terms, Individual Entities, and Relations, available online at: http://www.nactem.ac.uk/files/phatfile/cafetiere-report.pdf

[26] F. Ciravegna, A. Dingli, D. Petrelli, Y. Wilks, User-system cooperation in document annotation based on information, *Proceedings of the 13th International Conference on Knowledge Engineering and KM (EKAW02)*, Sigüenza, Spain, 2002

[27] M. O'Connor, H. Knublauch, S. Tu, B. Grosof, M. Dean, W. Grosso, M. Musen: Supporting Rule System Interoperability on the Semantic Web with SWRL, *4th International Semantic Web Conference (ISWC'05)*, Galway, Ireland, pp. 974-986, 2005

[28] Chmiel, K. et al: Testing the Efficiency of JADE Agent Platform, *3rd International Symposium on Parallel and Distributed Computing*, Cork, Ireland, 2004

[29] V. Kolovski, B. Parsia, E. Sirin: Extending the $\mathcal{SHOIQ}(\mathcal{D})$ Tableaux with DL-safe Rules: First Results, *International Workshop on Description Logic (DL-2006)*, 2006

[30] A. Uszok, J.M. Bradshaw, M. Johnson, R. Jeffers, A. Tate, J. Dalton, S. Aitken: KAoS Policy Management for Semantic Web Services, *IEEE Intelligent Systems*, 19(4), pp. 32-41, 2004

[31] L. Kagal, T. Finin, A. Johshi: A Policy Language for Pervasive Computing Environment. *4th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'03)*, Lake Como, Italy, 2003

[32] Niles, I. and Pease, A. Towards a Standard Upper Ontology. *2nd International Conference on Formal Ontology in Information Systems (FOIS'01)*, 2001

[33] H. Chen, T. Finin, and A. Joshi. Semantic Web in in the Context Broker Architecture. *2nd Annual IEEE International Conference on Pervasive Computer and Communications*, March 2004

[34] R. Masuoka, B. Parsia, Y. Labrou: Task Computing - the Semantic Web meets Pervasive Computing, *2nd International Semantic Web Conference (ISWC'03)*, Sanibel Island, Florida, USA, 2003

[35] M. Roman, C. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell, K. Nahrstedt: Gaia: a middleware platform for active spaces, *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 6, issue 4, 2002

[36] Juan Ignacio Vazquez, Diego López de Ipiña and Iñigo Sedano: SOAM: An Environment Adaptation Model for the Pervasive Semantic Web. *2nd Ubiquitous Web Systems and Intelligence Workshop (UWSI'06)*, pp.108-117, Glasgow, U.K., 2006

[37] H. Chen, F. Perich, T. Finin, A. Joshi: SOUPA: standard ontology for ubiquitous and pervasive applications, *1st International Conference on Mobile and Ubiquitous Systems: Networking and Services*, pp. 258 – 267, 2004