# A Service-Oriented Middleware for Providing Context Awareness and Notification

Luiz Olavo Bonino da Silva Santos
University of Twente
P.O.Box 217
7500AE – Enschede – the Netherlands
+31 53 489 44 54
l.o.bonino@ewi.utwente.nl

Peter Vink
Philips Research
P.O. Box WB61
5656AA – Eindhoven – the Netherlands
+31 40 274 95 52
peter.vink@tass.nl

Remco Poortinga–van Wijnen
Telematica Instituut
P.O. Box 589
7500AN – Enschede – the Netherlands
+31 53 485 04 92
remco.poortinga@telin.nl

## ABSTRACT
In the last few years context awareness has emerged as an important element in distributed computing. It offers mechanisms that allow applications to be aware of their environment and enable these applications to adjust their behavior to the current context. Considering the dynamic nature of context, the data flow of relevant contextual information can be significant. In order to keep track of this information flow, a flexible service mechanism should be available for the client applications. In this document we propose a demonstration of the Awareness and Notification Service (ANS) middleware. ANS provides context-awareness capabilities to users and client applications. The conditions for the notification and the notification itself are defined in rules that users submit to the service by means of a convenient rule language.

## Keywords
Context awareness, middleware, software demonstration.

## 1. INTRODUCTION
Context awareness represents an important use of distributed computing and introduces a new class of smart applications. Awareness of a user's surroundings and state helps applications to adapt their functionality depending on context changes and without direct user interaction. However, introducing context-awareness in applications demands a series of features such as discovery and selection of context sources, interaction with these sources, and manipulation and interpretation of contextual information, amongst others. These factors complicate the introduction of ad-hoc context-awareness solutions for system designers and developers. To tackle these requirements, a flexible mechanism allowing user applications to easily specify the relevant changes in the environment is of need.

Commonly, context-aware systems involve the interaction of distributed, mobile and heterogeneous applications and devices. Therefore, the use of concepts and technologies of Service-Oriented Computing can support tackling these issues of distribution, mobility and heterogeneity.


*MC'07*, November 26–30, 2007, Newport Beach, CA, Copyright 2007 ACM ISBN 978-1-59593-935-7//07/11…$5.00.

In this proposal we present an Awareness and Notification Service – ANS – following a rule-based approach which provides notifications depending on users' context. Section 2 introduces ANS. Section 3 details the proposed demonstration and its requirements.

## 2. THE AWARENESS AND NOTIFICATION SERVICE
The Awareness and Notification Service supports developers in adding context-awareness capabilities to their applications. Thus developers do not have to deal with monitoring, controlling and managing contextual information inside their applications. This avoids the necessity of creating specific context-awareness features for each application and, therefore fostering a rapid development. Applications are only responsible for registering so-called *monitoring rules*. These rules specify the context to be monitored and the notification to be sent once the expected context holds.

Once the client application has subscribed the monitoring rule, ANS starts gathering the required contextual information. In the case that the triggering condition contained in the monitoring rule holds, ANS proceeds to notify the client application according to the notification message specified in the rule. An example of such rule specifies that John should be notified when his children arrive at home. In this example, the ANS monitors the location of John's children and notifies him when they enter home.

Our approach considers that changes in the application's environment are modeled by means of Event-Condition-Action (ECA) rules [1][2]. A domain specific language has been developed to define context and context events supporting the specification of context-based reactive behaviors.

Following the Event-Control-Action pattern described in [1], three main parts are present in ARS as depicted in Figure 1. *EventMonitor* receives context data events from context sources through the Context Management Service (CMS). *EventMonitor* sends these events to the *Controller* that monitors them and evaluates the registered rules. If the triggering condition of the rule is evaluated true, *Notifier* is triggered to perform the suitable action. The action also depends on the user's context, for instance, if the user is in a meeting the notification can be sent via SMS to his mobile phone instead of via e-mail which would be the case if he were in his office. The subscribed rules and the ontologies used in ANS are stored in a *KnowledgeRepository* and made available for both *RuleManager* and *EventMonitor*.

The architectural design of ANS follows the Service-Oriented Architecture (SOA) principles. The service is implemented as a web service relying on standards such as SOAP, WSDL, UDDI and XML. The external entities with which ANS interacts are also implemented as web services, such as the client applications and the CMS. In the internal perspective, the ANS implementation follows the OSGi component based framework approach [4]. The current implementation of ARS uses the Oscar OSGi Framework [5].

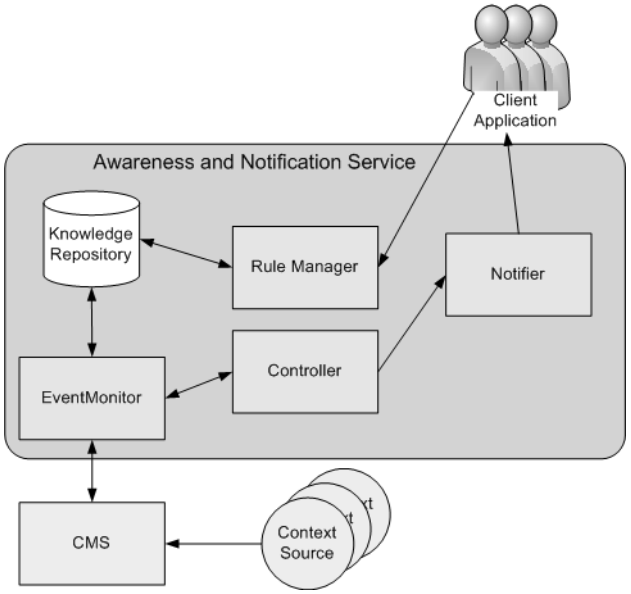


**Figure 1. The ANS architecture**

ANS exports two interfaces available both in the Oscar framework and as web service's interfaces through WSDL.:

• *IManageRule*, used by client applications to manage rules, and;

• *IReceiveContext*, which is a call back mechanism to receive information from context sources through the CMS. In addition, ANS reacts to Web Service events generate by CMS.

*RuleManager* is externally accessed via the *IManageRule* interface. The *RuleManager* provides facilities for unsubscribing, updating, starting and stopping rules. When a client application wants to register a rule, it sends the rule to the *RuleManager* that is responsible for parsing, validating and storing the incoming rule. In the parsing and validating phases, the *RuleManager* translates entered user rules to reaction rules that can be handled by the *Controller*. The rules received by the *RuleManager* from client applications are expressed in the ANS domain-specific ECA language called ECA-DL [6]. *RuleManager* transforms this ECA-DL rule into a rule that can be handled by the underlying rule-engine. Currently, ARS uses the JESS rule engine [10].

## 3. ANS DEMONSTRATION

The demonstration of ANS shows the awareness capabilities of the middleware by using a location-based scenario. To illustrate the use of the middleware we present a context source connected to an RFID sensor that informs the location of users. The users are identified by carrying RFID tags.

To demonstrate rules that identify when users enter certain rooms we use the signal strength of the RFID tags to create virtual areas of proximity. Since the signal strength of the RFID change depending on the distance to the reader, we define that certain ranges of the signal strength correspond to concentric regions as depicted in Figure 2. The RFID context source detects the signal strength of the tags and informs CMS. When the location-based rules are subscribed, ANS requests CMS to be informed whenever there is a location-based event. For each event CMS sends to ANS, the *Controller* component evaluates the rules and, if the user enters the region specified in the rule a notification is fired to the client.

In order to present the demonstration, requirements are: a table to place the computer running the middleware and the RFID reader and a space in front of the table of at least 3 meters. Visitors can carry the tags and participate in the demonstration. Moreover, a context source simulator is available to demonstrate more complex situations. The strength of the rule-based system is clearer when more complex rules are defined. For instance, combining multiple context sources by using the context source simulator together with the RFID context source allows the demonstration of a rule that evaluates the location of a user and the temperature of the environment.

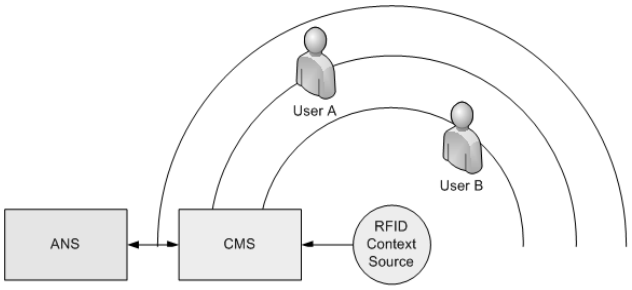


**Figure 2. Demonstration scenario**

## 4. ACKNOWLEDGMENTS

The work reported here is supported by the European Commission as part of the IST-IP Amigo project under contract IST-004182.

## 5. REFERENCES

[1] Dockhorn Costa, P., Pires, L. F., Sinderen, M. Architectural Patterns for Context-Aware Services Platforms. in Proceedings of the Second International Workshop on Ubiquitous Computing (IWUC 2005), Miami, May 2005, pp 3-19.

[2] Ipina, D., Katsiri, E. An ECA Rule-Matching Service for Simpler Development of Reactive Applications. Published as a supplement to the Proc. of Middleware 2001 at IEEE Distributed Systems Online, Vol. 2, No. 7, November 2001.

[3] Bonino da Silva Santos, L.O., Ramparany, F., Dockhorn Costa, P., Vink, P., Etter, R., Broens, T. A Service Architecture for Context Awareness and Reaction Provisioning. In Proceedings of the 2nd Modeling, Design, and Analysis for Service-Oriented Architecture Workshop (MDA4SOA 2007), Salt Lake City, USA, July 13th 2007.

[4] OSGi Consortium, http://www.osgi.org.

[5] Oscar OSGi Framework - http://forge.objectweb.org/projects/oscar/

[6] Dockhorn Costa, P., Ferreira Pires, L., van Sinderen, M., Broens, T., Controlling Services in a Mobile Context-Aware Infrastructure, in Proceedings of the Second Workshop on Context Awareness for Proactive Systems – CAPS 2006, Kassel, Germany, June 2000.