Journal of Applied Computing Research, 3(1):19-33
January-June 2013
© 2013 by Unisinos - doi: 10.4013/jacr.2013.31.03

Toward a distributed architecture for context awareness in ubiquitous computing

João Lopes
Universidade Federal do Rio Grande do Sul. Av. Bento Gonçalves, 950, 91501-970, Porto Alegre, RS, Brazil
jlblopes@inf.ufrgs.br

Márcia Gusmão, Cauê Duarte, Patrícia Davet
Universidade Federal de Pelotas. Rua Gomes Carneiro, 1, 96010-610, Pelotas, RS, Brazil
mzgusmao@inf.ufpel.edu.br, cduarte@inf.ufpel.edu.br, pdavet@inf.ufpel.edu.br

Rodrigo Souza
Universidade Federal do Rio Grande do Sul. Av. Bento Gonçalves, 950, 91501-970, Porto Alegre, RS, Brazil
rssouza@inf.ufrgs.br

Ana Pernas, Adenauer Yamin
Universidade Federal de Pelotas. Rua Gomes Carneiro, 1, 96010-610, Pelotas, RS, Brazil
marilza@inf.ufpel.edu.br, adenauer@inf.ufpel.edu.br

Cláudio Geyer
Universidade Federal do Rio Grande do Sul. Av. Bento Gonçalves, 950, 91501-970, Porto Alegre, RS, Brazil
geyer@inf.ufrgs.br

Abstract. The applications in Ubiquitous Computing (UbiComp) environments must be aware of their contexts of interest and adapt to changes in them. Thus, a major research challenge in the area of UbiComp is related to context awareness. Considering the high distribution, heterogeneity, dynamism, and mobility of ubiquitous environments, this paper presents an architectural model for context awareness, called EXEHDA-UC (Execution Environment for Highly Distributed Applications - Ubiquitous Context awareness). The proposal includes elements to support contextual data acquisition, actuation in the environment, and processing of contextual information. We consider that the main contribution of this work is an architecture that supports the managing of the acquisition, storage, and processing of context data, in a distributed way, independently of the application, in an autonomic and rule-based perspective. To assess the functionalities of the EXEHDA-UC, we present a case study, highlighting the prototypes developed, technologies employed, and tests carried out.

Keywords: ubiquitous computing, context awareness, distributed architecture.

Introduction

Mark Weiser's classic article (Weiser, 1991), considered the precursor of Ubiquitous Computing, describes the basic assumptions of this computational paradigm: ubiquity and transparency. These assumptions generate several challenges related to user access to computing environment, anywhere, any time, with any device, non-intrusively, keeping the users' focus on their activities. In this perspective, computational systems must interact in an autonomic way, no matter where the user is, constituting a highly distributed, heterogeneous, dynamic and mobile environment (Costa *et al.*, 2008; Caceres and Friday, 2012).

In this sense, one of the main research problems in the area of UbiComp is context awareness, which refers to the ability of applications to make changes in the characteristics of the ubiquitous environment, which are of its interest, and respond to these changes through an adaptation process (Silva *et al.*, 2012). This class of computational systems, reactive to the context, opens perspectives for the development of richer, elaborate and complex applications, exploring the dynamism of modern computational infrastructures and user mobility (Kakousis *et al.*, 2010).

The literature review indicates several challenges in the support of context awareness for ubiquitous applications, including: (i) context acquisition from distributed and heterogeneous sources, (ii) context processing and actuation in the environment, and (iii) context dissemination to interested users in a distributed and timely way (Bettini *et al.*, 2010; Bellavista *et al.*, 2012; Knappmeyer *et al.*, 2013).

The main contribution of this work is the proposal of a software architecture that offers to applications support for acquisition, storage, and processing of contextual information, as well as for the actuation in the environment, in a distributed way, independent of applications, in a rule-based autonomic perspective, and with support to mobility. Considering that context awareness service of EXEHDA middleware (Lopes *et al.*, 2012) does not address these features, EXEHDA-UC contributes particularly to the Context Recognition and Adaptation Subsystem of EXEHDA.

The paper is organized as follows. The second section presents an overview of EXEHDA middleware services. The third section presents an overview of EXEHDA-UC proposal. The fourth section describes the software architecture proposed for EXEHDA-UC. The fifth section presents a case study and an evaluation of the applications. The sixth section discusses the related work. Finally, the last section presents the concluding remarks.

EXEHDA middleware overview

EXEHDA middleware was conceived to support the execution of ubiquitous applications.The main properties of EXEHDA applications are: distributed, mobile, adaptive and reactive to the context. EXEHDA is composed of several integrated services that are conceptually organized in subsystems: data and code ubiquitous access, uncoupled spatial and temporal communication, large-scale distribution, context recognition and adaptation. The subsystem integration is shown in Figure 1 (Augustin *et al.*, 2008).

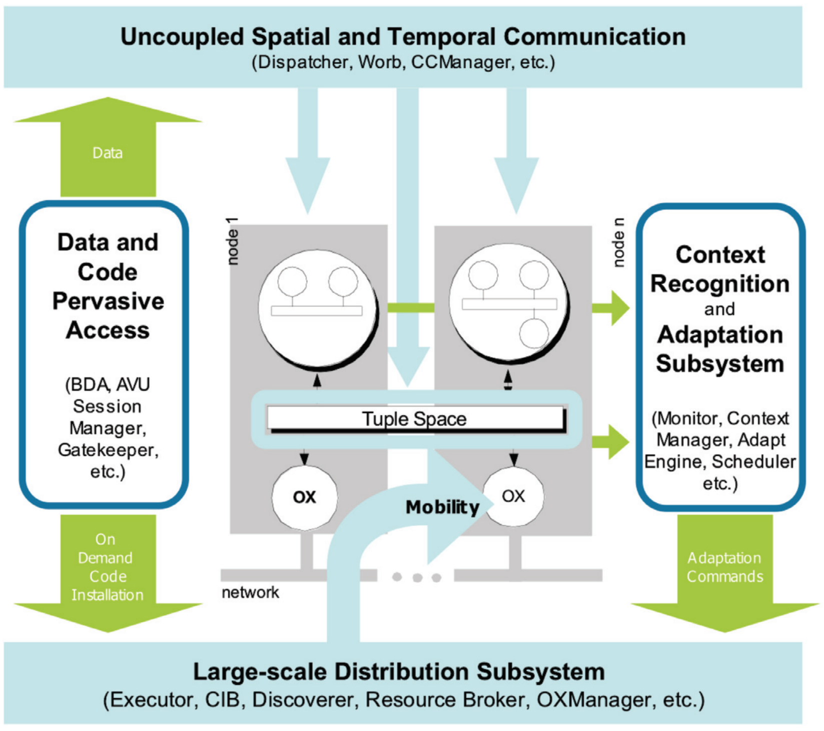

Uncoupled Spatial and Temporal Communication
(Dispatcher, Worb, CCManager, etc.)
Data
Data and Code Pervasive Access
(BDA, AVU Session Manager, Gatekeeper, etc.)
On Demand Code Installation
node 1
node n
Tuple Space
OX
OX
Mobility
network
Context Recognition and Adaptation Subsystem
(Monitor, Context Manager, Adapt Engine, Scheduler etc.)
Adaptation Commands
Large-scale Distribution Subsystem
(Executor, CIB, Discoverer, Resource Broker, OXManager, etc.)

Figure 1. EXEHDA subsystems.

Regarding communications, EXEHDA currently provides, through the Dispatcher, WORB, and CCManager services, three types of communication primitives, each one addressing a distinct abstraction level.

The Dispatcher Service corresponds to the lowest abstraction level, providing message-based communications. Message delivering is done through per-application channels, which may be configured to ensure several levels of protection for the data being transmitted. Protection levels range from data integrity, using digital signatures, to privacy through encryption mechanisms. Additionally, the Dispatcher Service uses a checkpointing/recovery mechanism for the channels, which is activated when a planned disconnection is in course. This feature may or may not be activated by the upper communication layers depending on its particular demands.

In order to make the development of distributed services easier, EXEHDA also provides an intermediary solution for communications, based on Remote Method Invocations, through the WORB Service. The programming model is similar to Java RMI, but optimized to ubiquitous requirements. More specifically, WORB remote method invocations, differently from Java RMI, do not require that the device keep connected during the entire execution of the method on the remote node. Instead, WORB was built on the functionality provided by the Dispatcher Service, including a per-invocation ID. The invocation ID remains valid during the disconnection, allowing the WORB to re-sync with the remote node after reconnection and obtain the returned values from the invocation.

At a higher level, the CCManager Service provides tuple-space based communications. It builds on the WORB Service, which also handles planned disconnections, providing to applications an anonymous and asynchronous communication support. This model is provided in the CCManager Service and is better suited to scenarios in which application components might migrate among nodes, since it does not require both sides to coexist for the communication to take place.

From the middleware point of view, environment resources fit in one of two categories: processing node or specialized resources. The former corresponds to the nodes, which effectively execute and whose access is managed by the middleware. The latter corresponds to specialized devices, e.g. printers, scanners, etc., whose access is not done through one of the middleware services, but through the use of some specific libraries. Although not managed by EXEHDA, the specialized devices are also cataloged in the CIB Service in order to allow applications to locate and use them.

The Discoverer Service is in charge of finding specialized resources in the environment based on an abstract definition of the resource. Typically, this service interacts with the CIB Service from its own cell, aiming at satisfying the resource discovery request in the scope of the local cell. When the local resources fail to fulfill the request, the Discoverer Service interacts with the Resource Broker service of the neighbors' cells. The strategy adopted in this extra-cell search is characteristic of the particular Discoverer Service instance in use. These services employ a language to describe resources and its interfaces are standardized. Since the middleware does not manage specialized resources, the results of a Discoverer Service search do not imply resource allocation or even resource reservation.

The Monitor Service implements a monitoring scheme based on sensors, which employs indexes to describe specific aspect of the environment. These sensors can be customized through parameters. The whole set of sensors installed on a node is part of the node description information registered in the CIB Service. The data generated by each sensor is gathered by the Monitor Service, which typically runs on the same node where sensors are installed. The gathered data is published by the Monitor Service to a Collector Service, which typically runs on the base-node.

Both the gathering of data by the sensors and the publication to the Collector Service by the Monitor occurs in discrete multiples of a per-node configured quantum. The quantum parameter allows the resource owner to control, externally to the middleware, the degree of intrusion of the monitoring mechanism in the host. After a quantum of time expires, the Monitor Service executes a pooling operation over the active sensors in the node. Then, it applies the publishing criteria specified for the sensor data, determining, or not, the generation of a publishing event for that sensor. Thus, the events generated after a quantum expiration are grouped into a single message, reducing the amount of data that the Monitor has to transmit to the Collector.The Collector Service aggregates information from several monitors in the cell and forwards them to the registered consumers.

EXEHDA-UC proposal overview

The model proposed for EXEHDA-UC provides dynamic handling of context information, while the ubiquitous applications are been executed. For this, in EXEHDA-UC we employed a rule-based and event-driven approach that enables the possibility of associating processing rules to the contexts of interest, which can be triggered automatically whenever an event occurs. The event, in our view, relates to changes in the context status. Figure 2 illustrates the approach proposed for EXEHDA-UC.

The autonomic approach is also a key aspect in the design of the proposal, corresponding to the ability to construct contexts independently of the application execution, which can be composed by the context data obtained from different sensors distributed in cells forming the ubiquitous environment.

A usage scenario for this approach is described below and corresponds to an application related to LDAS - Didactic Laboratory of Seed Analysis:

- *Application:* Control of LDAS Physical Environment Condition.
- *Component:* Temperature Control in the Seed Germinators Room.
- *Context of Interest*: temperature.
- *States of Context:* temperature values collected.
- *Sensors:* digital temperature sensor based on the 1-wire technology.
- *Event:* temperature outside the boundary (>= 27º C or <= 15º C).
- *Rule:*

  rule “Warning Light”
  when
  Verify(LabRoom.Temperature >= 27 or LabRoom.Temperature <= 15)
  then
  Activate(LabRoom.Light)
  end
- *Actuation:* addressable electronic key based on the 1-wire technology (the rule triggers the software component that activates a warning light in the room of the person in charge for LDAS).

EXEHDA-UC Software Architecture

The EXEHDA-UC encompasses two types of servers: Border Server, responsible for the interaction with the environment through sensors and actuators, and Context Server, responsible for processing the contextual information. These servers are located in cells of the ubiquitous environment managed by EXEHDA, where each cell has one Context Server and can contain several Border Servers.

The proposed architecture for EXEHDA-UC enables communication: (i) among Border Servers and Context Servers, (ii) among Con-

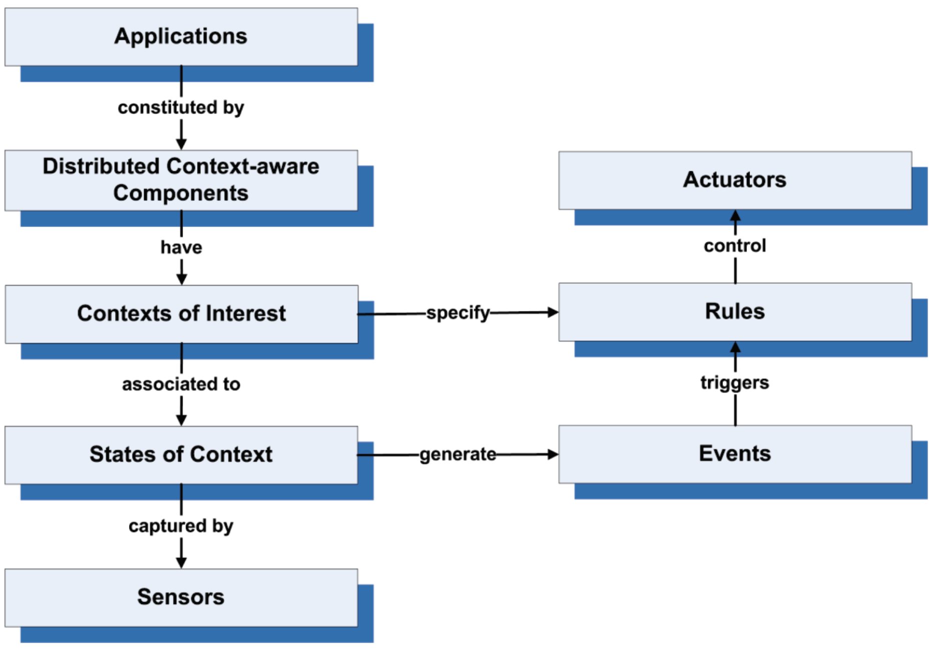


Figure 2. Rule-based and event-driven approach proposed for EXEHDA-UC.

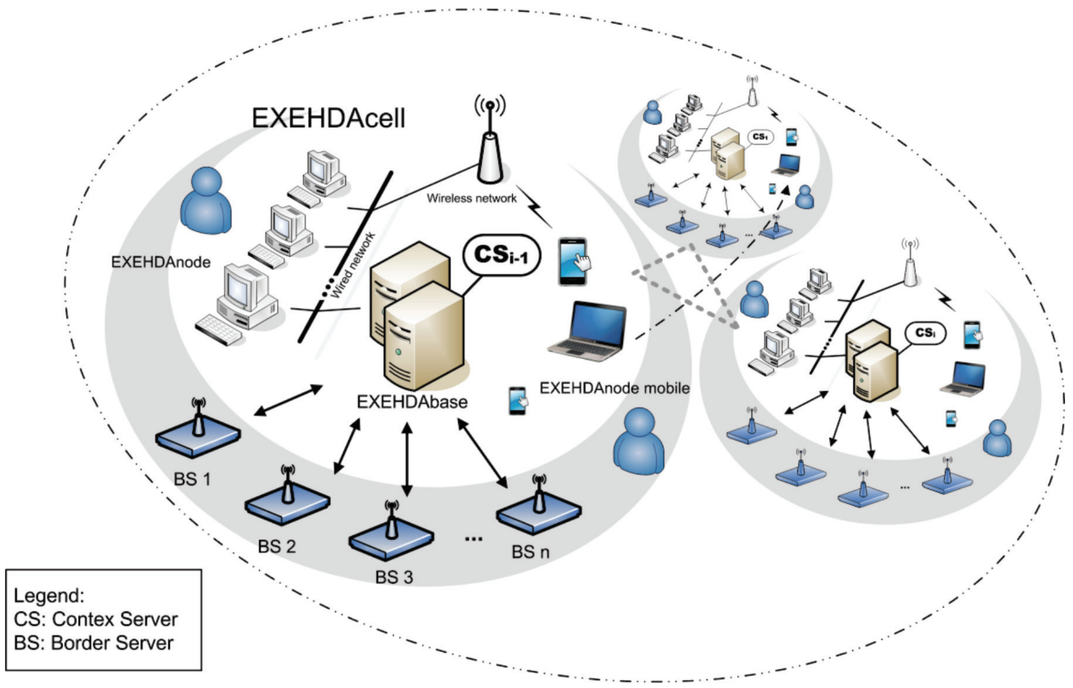


Figure 3. Ubiquitous environment.

text Servers located in different cells of ubiquitous environment managed by EXEHDA, and (iii) with other middleware services, or applications. An overview of this architecture is shown in Figure 3, in which the components are mapped over the ubiquitous environment.

The EXEHDA-UC architecture can acquire contextual information in an autonomic way, as well as allow remote actuation in the environment through electromechanical devices. In a ubiquitous environment different information is scattered, being necessary to be collected through sensors geographically distributed for the users to have access to the information. This collected data should also be stored for further processing by both middleware as applications.

The description of EXEHDA-UC, presented in the next sections, is organized based on it servers and corresponding features, and the necessary associations between them, and with the other services of the EXEHDA middleware carried out.

Border server

The proposed architecture for the Border Server includes three modules targeted to: (i) manage sensor networks, (ii) make publications, and (iii) manage actuator networks. A detailed view of the architecture is shown in Figure 4.

Sensing Module provides handling of sensor networks, enabling the individualization of processing by sensor. It covers aspects from physical management (interfaces, reading frequency) to computational normalization (validation, translation) of the collected values. It also enables publication of the information collected by sensor networks in Context Server. This module comprises six components, which are described below.

Operating Parameters of Sensors Component specifies the driver to be used for reading different sensors, as well as the agenda to data acquisition. This agenda handles sensors individually, allowing the specification of periodic readings, linked to dates and/or specific times. This component is defined by the user in the Context Server, using a YAML language (http://www.yaml.org). The specifications available in the Repository of Context are used and, once ready, the YAML file is transferred to the Border Server.

Based on the Operating Parameters of Sensors, the *Scheduler Component* triggers readings, considering the user's application interests. The Scheduler uses the clock of the Border Server to trigger its procedures, being possible to specify the frequency of occurrence. Each activation triggers a YAML parser to interpret the sensor which should be read and activated by the corresponding driver.

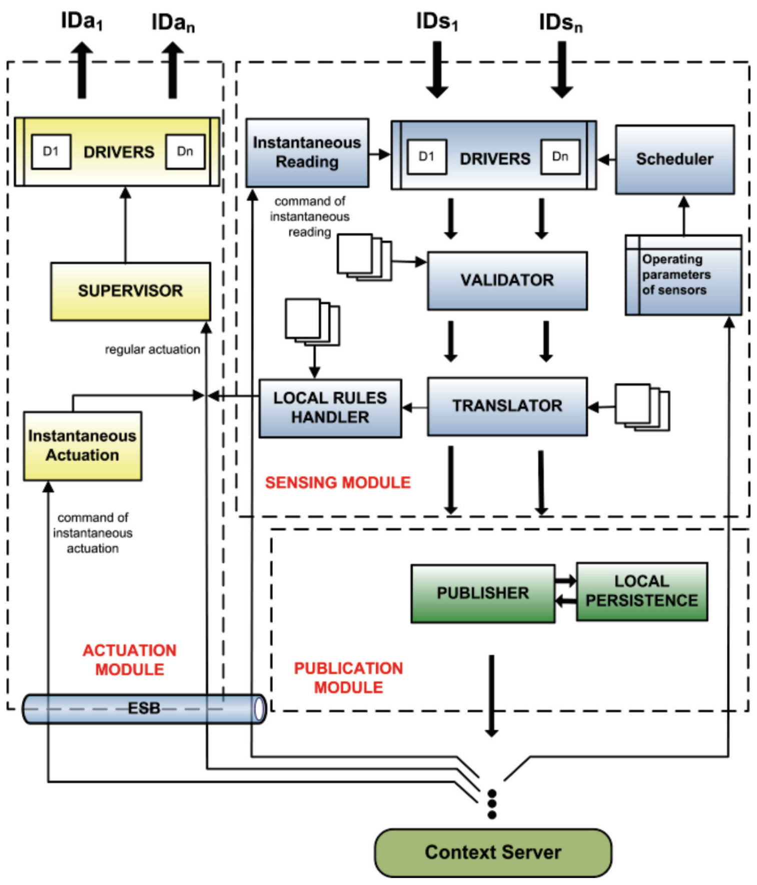


Figure 4. Border server.

The sensor's *driver* is responsible for the acquisition of physical value measured by the sensor. It is usual that each type of sensor has a specific form of access to data acquisition. The strategy of encapsulating the driver operation aims to prevent that the operational differences of each driver may have an impact in the other layers of software architecture. Among other aspects this encapsulation improves the maintenance (replacement of sensors, updating drivers). The driver of each sensor is individualized by an identification code, allowing the addition of a new driver or the modification of an old driver, without further specification.

The *Validator Component* assesses whether a particular data collection should be published or not, using criteria specified by user-defined rules. These criteria are based on rules, for example, a rule may state that only values whose variance is greater than 5% from the previous reading should be published, or values that are in a certain range should be published. The rule is identified in the *Operating Parameters of Sensors Component*, considering the sensor ID (IDs).

The *Translator Component* performs the adequacy of the collected data to the nature of the user application, also by a rule. For example, temperature ranges can be converted into descriptions such as "High", "Medium", "Low", through a procedural rule. This component helps to optimize the volume of data transferred between servers, also increasing the readability of the collected data.

The *Instantaneous Reading Component* enables a sensor reading, considering the application's demand at any time. The requests can come from user's applications or Context Servers, as a result of rules implementation. It employs an Enterprise Service Bus (ESB) for receiving asynchronous requests and, considering the sensor ID, triggers the corresponding driver.

The *Local Rules Handler Component* processes contingency rules intended to prevent the devices involved from reaching critical states. These rules act on the management mechanism of actuation, enabling or disabling actuators. This component is activated whenever the reading of a particular sensor occurs, acquiring high importance when the communication with the Context Server is interrupted by problems in the network infrastructure.

The *Publication Module* is responsible for coordinating the main data flow between Border Servers and the Context Server, promoting the publication of all collected data and ensuring a *Local Persistence* in the periods when the publication is frustrated. This module is composed by *Publisher Component*, which interoperates with the Acquisition Module of the Context Server, performing submissions of collected data. These submissions are made using the ESB of the Acquisition Module, individualized by application. Depending on the nature of the application, for security, data are transferred in an encrypted way.

The *Actuation Module* is responsible for managing the actuators. This module is composed by the *Instantaneous Actuation Component*, which receives asynchronous commands at any time, via an ESB. These commands are originated from the Context Server, as a result of the execution of a rule, as well as from a user application. The parameters used are the ID of the actuator and the corresponding operation patterns (duration, power activation, etc...), which are passed to the *Supervisor Component* for processing. The *Supervisor* binds actuation commands from three different sources: the regular actuation, the instant actuation, and those from the *Local Rules Handler*. Once the *Supervisor* has received the parameters to control the actuation, it can activate the required driver for the actuator that is being managed.

In order to identify anomalous behaviour related to the management of the actuators, the *Supervisor* has a specification for actuator (IDa) of how many times for a unit of time transitions in the state of the actuator are expected to occur. The objective is to identify potential conflicts between rules in the Context Server and rules of contingency in the *Local Rules Handler Component*, as well as unconformities between instant actuation commands, triggered by the user, and active rules in the servers. Also, this component handles the parameters for activating the actuators, implementing through drivers the procedures of activation and deactivation, control of operational power, validity time of actuation, among others.

The actuator's *Drivers* have a similar purpose to the sensor's *Drivers*, i.e., they encapsulate the procedures specific to each actuator, most often employing libraries and/or software provided by the manufacturers. This approach preserves the spread of implementation aspects to the upper layers of the Border Server's architecture.

Context server

The modules of Context Server, following described, interoperate in the provision of the functionalities for context awareness services. Each of these modules is responsible for one stage of context awareness, since its acquisition until the time that it is stored and/or passed on to anyone who requested the contextual information. An overview of the Context Server architecture is illustrated in Figure 5, characterizing the relationship with Border Servers, other middleware services, other remote Context Servers, and applications.

Acquisition module

It provides support for the capture of contextual information, collected by Border Servers, considering logic (software interfaces) and/or hardware sensors. This module presents a server behaviour whose functionality is implemented through an ESB, allowing Border Servers, whenever there are significant variations in contextual data, to publish these data.

Actuation module

It is in charge of the actuators control (activation, deactivation, and configuration), after being notified by other Context Server modules. This module receives the actuator's identifier and operational parameters to be used, and interoperates with Border Servers for triggering the actuators. In general, the Actuation Module is responsible for triggering the ubiquitous environment actions that change the state of the environment, enabling the use of context awareness in applications for automation and control.

Notification module

It deals with notifying the result of context processing performed by the Interpretation Module. This module receives, through the Communication Module, subscriptions from all services and/or applications that require notifications about the context state. The Notification Module also receives all decisions of

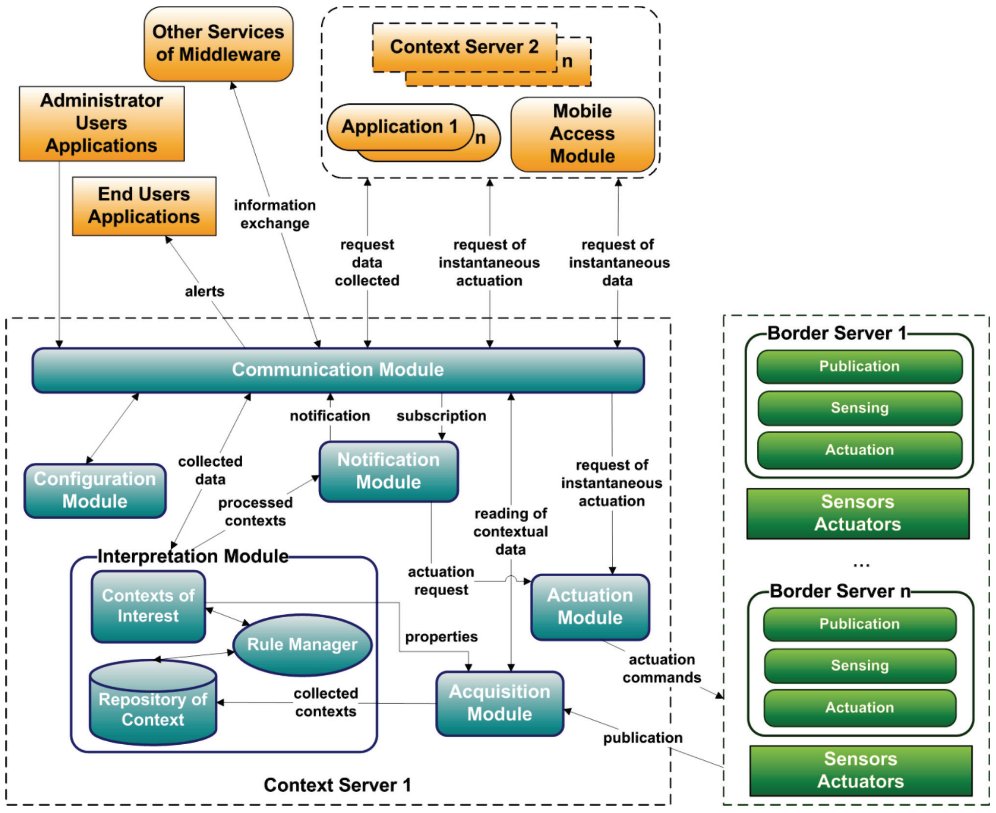


Figure 5. Context Server.

actuation, resulting from the autonomic treatment of context rules.

Communication module

This module is used by remote Context Servers and/or applications to request contextual data and/or to trigger actuators. This module provides the dissemination of context information to other middleware services, as well as sends messages to users. It receives requests through an ESB, and interoperates with other architectural components using messages, as well as with users using public protocols for sending messages over the cellular network (SMS - Short Message Service), Google Talk, and emails. The module includes a repository responsible for storing information necessary for sending alerts.

Configuration module

It allows the management of Context Server settings, including specifications of sensors and actuators, as well as information of equipment in which the context is being collected.

Mobile access module

This module provides mobile access to EXEHDA-UC. It is organized in two blocks (Figure 6): Block A displays contextual information, and Block B displays proactive alerts. Particularly, the provision of proactive alerts on a hardware platform that can follow the user, while he/she performs its activities in different places, maximizes the ubiquity of the proposed context awareness service.

Block A – Display of Contextual Information - provides graphical and textual reports for users, considering their contexts of interest. Communication with the Context Server happens employing a two-step protocol. In the first an inspection of the context information that is being treated is made. In the second, data are requested for specific contextual information in a time interval. The features of this block are organized into three modules, as follows:

Report Module: this module displays contextual information on the mobile device. The main operations available through its interface are: (i) to request the list of contextual information which is handled by the Context Server, (ii) to

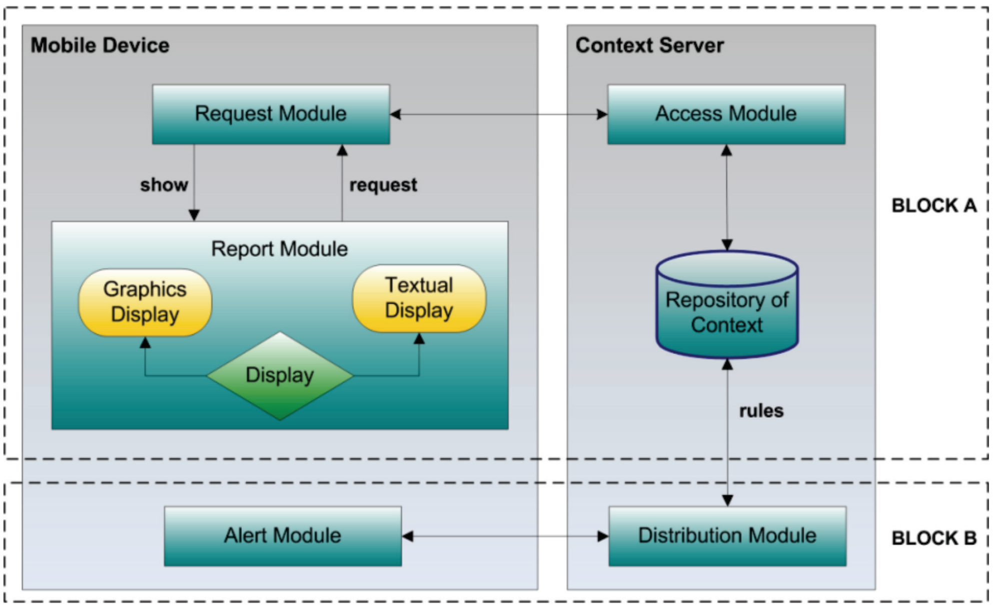


Figure 6. Mobile access manager.

select contextual information that is displayed in the report, (iii) to select the report type: graphical or textual, and (iv) to enable for the user, once the report is displayed, alternatives to customization of the period of data shown.

Access Module: this module consists of a service that runs in the Context Server, having access to the Repository of Context. The features of this module are: (i) to return the list of contextual information managed by the Context Server, and (ii) to return the contextual information relevant to the user's context of interest, in a specific time interval.

Request Module: aims at requesting the contextual information to the Access Module, considering a demand of the Report Module. This module has three functions: (i) to request the list of all the contextual information handled by the Context Server, (ii) to request specific contextual information desired by the user in the specified time interval, (iii) to send this request to the Access Module through the ESB, and (iv) to receive and interpret contextual information from the Access Module, making it available to the Report Module.

Block B - Handling Proactive Alerts - is intended to notify the user about the occurrence of events of interest, through proactive notification. The functionalities of the block are organized into two modules, described as follows.

Alert Module: runs on mobile devices, providing alerts for users. For this, the native mechanism for notification of the mobile platform is used. This option provides for the user an integrated management of alerts on their mobile device. Its main functions are: (i) to recover, in the time interval specified by the user, alerts from the Distribution Module, and (ii) to provide for the user these alerts in the notification area of the mobile device.

Distribution Module: runs on the same equipment of the Context Server, under uninterrupted operation. Its main functions are: (i) to receive alerts produced by the Rule Manager Component of the Context Server, and (ii) to provide alerts to mobile devices. This module operates keeping the alerts produced by different contextual rules, treated in the Interpretation Module of the Context Server. This module is accessed by mobile devices through the Communication Module of the Context Server, which uses an ESB to provide access to different functionalities of the Context Server.

Case study

The EXEHDA-UC architecture is not specific to a particular problem domain, but is designed to be comprehensive, aiming to meet different domains. In this sense, despite the fact that the case study described in this section is related to a project in the area of agriculture, the architecture enables support to other application domains.

Thus, this section summarizes the main aspects of the case study related to the AMPLUS project (http://amplus.ufpel.edu.br), used to evaluate the functionalities of EXEHDA-UC. The case study includes tasks related to sensing, collection, processing, and notification of context. In this case study an application for a Web interface and mobile devices was developed, whose usability was evaluated by the users.

The main purpose of seed analysis is to determine the quality of a lot and thus its value for seeding. So that the objectives of the analysis are achieved it is necessary that laboratories have appropriate equipment and follow standardized methods and procedures (ISTA, 2012). In this way, AMPLUS project was conceived to provide mobile and context-aware services, allowing storage of contextual states that characterize the equipments of the Didactic Laboratory of Seed Analysis (LDAS - http://amplus.ufpel.edu.br/ldas), through the implementation of various tests, and a proactive actuation when necessary.

The scalability and robustness were the criteria to select the technologies used for the prototyping of the mechanisms for collecting, storing, and processing of context, as well as actuation in the environment. In this sense, the code of the Border and Context Servers is written in the Python language. The XML-RPC (Extensible Markup Language - Remote Procedure Call) (http://www.xmlrpc.com) is employed to implement the ESB, used for interoperability. The Repository of Context uses PostgreSQL for deployment of databases. The sensors and actuators communicate at 1-Wire protocol (http://ubiq.inf.ufpel.edu.br/1-wire).

In the perspective of EXEHDA-UC architecture, the process of defining the application begins with a survey of sensing demand, and corresponding physical design of the sensors network cabling. The sensors of AMPLUS project use the 1-Wire protocol for the physical management. Each sensor has a unique identification defined by the manufacturer, which is associated with a logical identification number, which is registered in the Repository of Contexts. The sensors are associated with contexts of interest defined by users, through their logical identification number. Each context of interest may be formed by one or more sensors. Associated with each context of interest there is a rule, which processes computationally the sensors involved with it.

This way, as the sensors are published by the Border Server, rules written in Python are triggered by the Context Server in an autonomic way.The reading of the sensors is done by drivers (software) for specific types of sensor, which are triggered by the Scheduler Component of the Border Server. In this case study, the Scheduler is activated every minute.

In order to facilitate handling by the user, the rules can be parameterized. For example, in the application of this case study, it is considered that the range of 40°C to 41°C is “high temperature” and between 18°C and 19°C is “low temperature” of a particular device used in AMPLUS Project. Still, it is necessary to register information about the infrastructure of sensing through the Configuration Module of the Context Server.

Finally, the end user application can be prototyped. This is influenced by context data notified through the modules of Notification and Communication, as well as through rules that trigger adaptive procedures. For example, an application of AMPLUS project that monitors the temperature of a seed germination chamber can subscribe to receive alerts when the temperature is outside a given range appropriate to the experiment. Associated with this rule, another rule states how the alert will be sent, for example, in opening hours a light alert appears in the technician laboratory room, otherwise an alert will be sent by email. To receive the alerts, persons in charge of infrastructure and/or equipment in particular need to be previously registered in the Repository of Context through the Configuration Module.

Developed application

The developed application uses two approaches, defined with users of the LDAS, one addressed to a Web interface and another to mobile devices.

The Web interface allows the selection of the context of interest to be displayed, providing a textual report with relevant data collected the previous week. Along with this report a menu that lets the user to select a graphic visualization of data as well as the creation of custom reporting are available.

The graphical report (Figure 7) provided by the system allows simultaneous viewing of information from multiple sensors. The selection of sensors is made from a menu that supports multiple selections. Also a resource inspection which allows comparison of values at a given moment of time is available. The time window of the data being displayed can be set by the user via the same graphical interface that displays the sensing values.

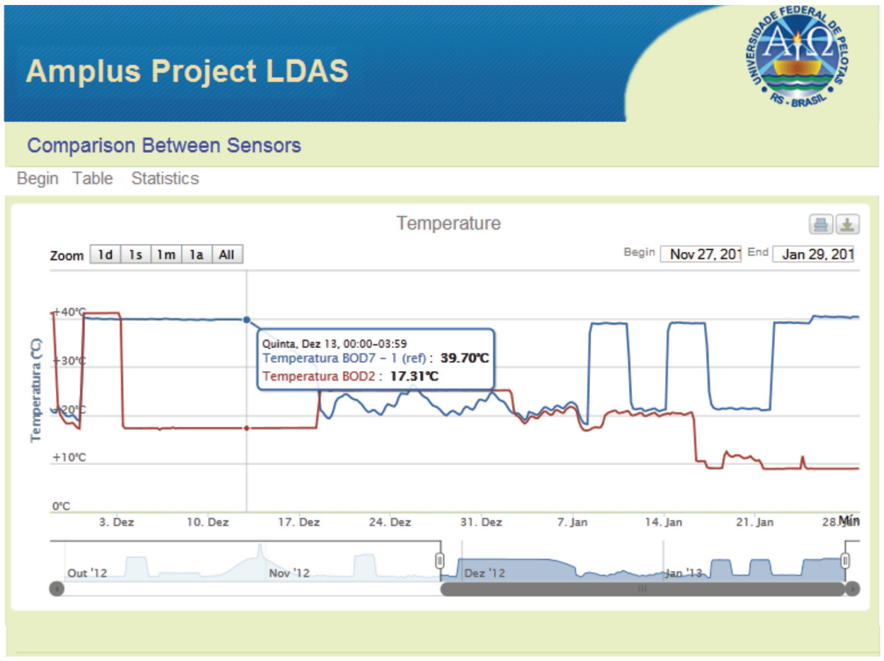
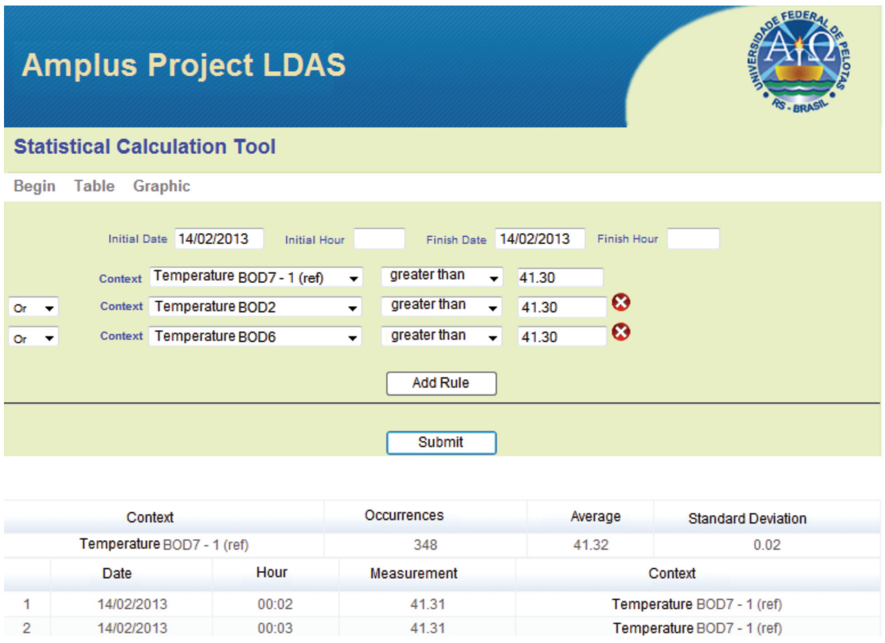


Figure 7. Graphical report.



Initial Date 14/02/2013 Initial Hour Finish Date 14/02/2013 Finish Hour

	Context	Temperature BOD7 - 1 (ref)	greater than	41.30
Or	Context	Temperature BOD2	greater than	41.30
Or	Context	Temperature BOD6	greater than	41.30

Add Rule

Submit

Context	Occurrences	Average	Standard Deviation
Temperature BOD7 - 1 (ref)	348	41.32	0.02

	Date	Hour	Measurement	Context
1	14/02/2013	00:02	41.31	Temperature BOD7 - 1 (ref)
2	14/02/2013	00:03	41.31	Temperature BOD7 - 1 (ref)

Figure 8. Contextual data crossing.

A feature that performs the contextual data crossing, involving multiple sensors from different rules, was designed to provide data for researches in LDAS. All manipulation is done through the Web interface with facilities for adding, removing, and editing rules and parameters, as shown in Figure 8.

Still, in order to promote the proactivity of the AMPLUS Project with the user community, we developed interfaces for communication services: e-mail and SMS to the cellular network. These messages are produced from the processing of contextual rules autonomously by the Interpretation Module of the Context Server.

The routine of the laboratory workers implies mobility in different physical environments of the LDAS. To address this situation, the Communication Module provides an interface for visual alerts, which are activated whenever a device is in a state that requires attention. Considering these alert, details can be inferred through the computer interface of the AMPLUS Project.

The interface for mobile access was intended to the Android platform. Through the initial interface the user can select the sensor to be displayed, either through a textual or graphical report. These reports enable the user to specify the display interval (hour, day, week), and the vertical axis adjustment is done automatically, minimizing the use of scrolling. The display of alerts is done through the interface provided by the Android platform itself. This aspect enhances the integration between the alerts mechanism and the functionalities of the user's smart phone. The interfaces corresponding to these features are shown in Figure 9.

Evaluation

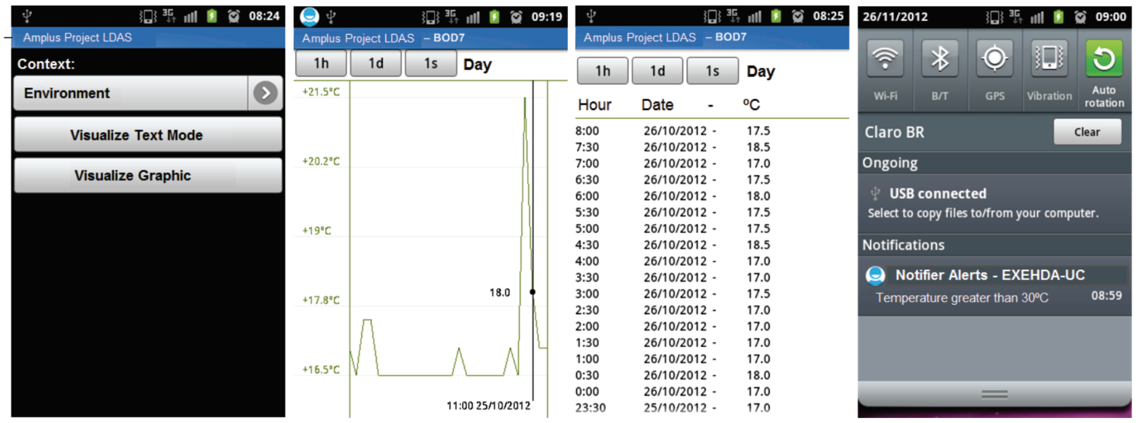
The literature states that prototyping is the most commonly applied technique to evaluate features of a middleware. The main target is to show the capabilities of the middleware based on experimental applications. The user experience can be explicitly evaluated, through questionnaires that assess the usability of the applications (Knappmeyer *et al.*, 2013).

Thus, this section presents the experiment details and the results obtained with the application's evaluation. The evaluation regards the application acceptance, which involved LDAS volunteer users. For this study we considered 4 teachers, 5 students, and 1 technician. Each participant used a mobile device and a desktop. We performed a basic training on the application operation beforehand. Participants were asked to use the application and respond to an evaluation questionnaire regarding the experience concerning the use of the system.

The answers should be within a range of five points, ranging from 1 point (totally disagree) to 5 points (totally agree). To evaluate the model acceptability, checking the system usability, the questionnaires were defined based on the Technology Acceptance Model (TAM) (Yoon and Kim, 2007). The TAM model considers the following main themes for application acceptance: (i) Ease of Use: the degree in which users evaluating the application may reduce their effort; (ii) Usefulness: the degree in which users evaluating the application may improve their performance.

Tables 1 and 2 contain the questionnaire applied to users and the answers obtained. The questions were designed in order to be simple, short and direct. Both tables present the question in the first column and the percentage obtained with the number of users in brackets following from "Totally Disagree" to "Totally Agree". The last column shows the consolidation average percentage score obtained by the responses, which varied between zero and five.

Analyzing the results it can be seen that approval is high for both ease of use and usefulness. However, there were results in the range "indifferent" in the last two issues of usefulness. This can be interpreted as a concern for the quality control of experiments developed in LDAS, which depends basically on the use of autonomic mechanisms, without the usual human intervention, for issuing alerts for contextual states that require immediate actuation. In this case, a strategy that can be adopt-



Hour	Date	-	ºC
8:00	26/10/2012	-	17.5
7:30	26/10/2012	-	18.5
7:00	26/10/2012	-	17.0
6:30	26/10/2012	-	17.5
6:00	26/10/2012	-	18.0
5:30	26/10/2012	-	17.5
5:00	26/10/2012	-	17.5
4:30	26/10/2012	-	18.5
4:00	26/10/2012	-	17.0
3:30	26/10/2012	-	17.0
3:00	26/10/2012	-	17.5
2:30	26/10/2012	-	17.0
2:00	26/10/2012	-	17.0
1:30	26/10/2012	-	17.0
1:00	26/10/2012	-	17.0
0:30	26/10/2012	-	18.0
0:00	26/10/2012	-	17.0
23:30	25/10/2012	-	17.0

USB connected
Select to copy files to/from your computer.

Notifier Alerts - EXEHDA-UC
Temperature greater than 30ºC 08:59

Figure 9. Interfaces of the mobile application.

Table 1. Ease of use evaluation.

Question	Totally disagree	Partially disagree	Neutral	Partially agree	Totally agree	Average
1. The application is easy to understand.	0,0% (0)	0,0% (0)	0,0% (0)	40,0% (4)	60,0% (6)	4,60
2. The application is easy to use.	0,0% (0)	0,0% (0)	0,0% (0)	30,0% (3)	70,0% (7)	4,70
3. The options are clear and objective.	0,0% (0)	0,0% (0)	10,0% (1)	20,0% (2)	70,0% (7)	4,60
4. I can select a context of interestwith little effort.	0,0% (0)	0,0% (0)	0,0% (0)	20,0% (2)	80,0% (8)	4,80
5. I can access the graphical reportswith little effort.	0,0% (0)	0,0% (0)	0,0% (0)	30,0% (3)	70,0% (7)	4,70

Table 2. Usefulness evaluation.

Question	Totally disagree	Partially disagree	Neutral	Partially agree	Totally agree	Average
1. The options presented are relevant.	0,0% (0)	0,0% (0)	0,0% (0)	30,0% (3)	70,0% (7)	4,70
2. The application makes it easy to obtain contextual data from multiple sensors.	0,0% (0)	0,0% (0)	0,0% (0)	40,0% (4)	60,0% (6)	4,60
3. The application makes my mobility easy.	0,0% (0)	0,0% (0)	0,0% (1)	40,0% (4)	60,0% (6)	4,60
4. The application makes immediate actuation easy, considering the sending of one alert or message.	0,0% (0)	0,0% (0)	30,0% (3)	30,0% (3)	40,0% (4)	4,10
5. I would use this application in my work at LDAS.	0,0% (0)	0,0% (0)	30,0% (3)	20,0% (2)	50,0% (5)	4,20

ed is to intensify the testing and validation with users and start a gradual deployment of applications.

Also, considering the results, we can state that the prototyping has shown promising results, both in terms of the middleware architecture design as the technologies used.

Related work

In this work we study the following related work: CARE (Agostini *et al.*, 2009), CoCA (Ejigu *et al.*, 2008), HiCon (Cho *et al.*, 2008), Solar (Chen *et al.*, 2008), and WComp (Ferry *et al.*, 2013). These projects were selected because they represent a substantial set of the research that has been done in recent years toward architectures for supporting context awareness.

The study has been done considering the main design assumptions of EXEHDA-UC: (i) distributed architecture, (ii) support for sensor and actuator networks, (iii) autonomic acquisition of context data, (iv) support for rule processing, and (v) support for distributed actuation.

The architectures studied did not maintain a decentralized approach for all stages of the context processing, which is not appropriate for the requirement of large scale distribution of ubiquitous environments. In turn, the architectural model of EXEHDA-UC is structured in a distributed way, at all stages of handling context information, from acquisition to actuation.

The EXEHDA-UC can manage sensor and actuator networks, optimizing the management of acquisition of context data from various types of sensors, usual in computational environments for providing ubiquitous applications. Such feature is found in part in the projects CoCA and HiCon, which have support to sensor networks. The project WComp allows actuation in the environment; however, it does not support actuator networks.

With the exception of the CARE and Solar projects, the others allow the use of specific mechanisms for acquisition, adopting a strategy of separation between the acquisition and use of context. Besides contemplating this aspect, the EXEHDA-UC presents a differential that consists in the employment of an autonomic approach in the collection of contextual data, as these continue to be obtained by the mechanism, even if the applications involved in their use are not in operation.

In most projects, the handling of rules is restricted to a few steps of the context processing. The EXEHDA-UC distinguished by its software architecture has been designed to support distributed processing of customizable rules, which can be linked to different treatment levels of contextual data, both in Border Servers as in Context Servers.

Concluding remarks

The EXEHDA-UC architecture provides a distributed approach in context acquisition and/or actuation in the environment. Furthermore, EXEHDA-UC approximates the sensor network to an infrastructure with autonomous capability of context treatment.

Enhancing the distributed approach in the treatment of contextual data, the EXEHDA-UC supports the concept of sensor and actuator networks, which provides aspects of modularity in software development and also contributes to the organization of procedures for creating and maintaining the networks involved.

The main contribution of EXEHDA-UC is the managing of the acquisition, storage, and processing of context data, independently of the application, in an autonomic and rule-based perspective. The autonomic approach and the use of rules in different parts of the distributed architecture of EXEHDA-UC is a substantial difference to the related work.

The usability evaluations, performed by users of the AMPLUS project, have been positive and have brought returns to the consolidation of EXEHDA-UC.

Among others, the following aspects should be explored in future works: (i) to explore case studies in which processing rules employ higher level inference mechanisms; and (ii) to use the EXEHDA-UC architecture for providing situation awareness, in which future situations of the different contexts in the ubiquitous environmentare anticipated.

References

AGOSTINI, A.; BETTINI, C.; RIBONI, D. 2009. Hybrid Reasoning in the CARE Middleware for Context Awareness. *International Journal of Web Engineering and Technology*, **5**(1):3-23. http://dx.doi.org/10.1504/IJWET.2009.025011

AUGUSTIN, I.; YAMIN, A.; SILVA, L. 2008. Building a smart environment at large-scale with a pervasive grid middleware. *In:* I. AUGUSTIN; A. YAMIN; L. SILVA, *Grid Computing Research Progress*. 1st ed., New York, Nova Science, p. 323-344.

BELLAVISTA, P.; CORRADI, A.; FANELLI, M.; FOSCHINI, L. 2012. A Survey of Context Data Distribution for Mobile Ubiquitous Systems. *ACM Computing Surveys*, **44**(4):24:1-24:45. http://dx.doi.org/10.1145/2333112.2333119

BETTINI C.; BRDICZKA O.; HENRICKSEN K.; INDULSKA J.; NICKLAS D.; RANGANATHAN A.; RIBONI D. 2010. A Survey of Context Modelling and Reasoning Techniques. *Pervasive and Mobile Computing*, **6**(2):161-180. http://dx.doi.org/10.1016/j.pmcj.2009.06.002

CACERES, R.; FRIDAY, A. 2012. Ubicomp Systems at 20: Progress, Opportunities, and Challenges. *IEEE Pervasive Computing*, **11**(1):14-21. http://dx.doi.org/10.1109/MPRV.2011.85

CHEN, G.; LI, M.; KOTZ, D. 2008. Data-Centric Middleware for Context-Aware Pervasive Computing. *Pervasive and Mobile Computing*, **4(**2):216-253. http://dx.doi.org/10.1016/j.pmcj.2007.10.001

CHO, K.; HWANG, I.; KANG, S.; KIM, B.; LEE, J.; LEE, S.; PARK, S.; SONG, J.; RHEE, Y. 2008. Hicon: a Hierarchical Context Monitoring and Composition Framework for Next-Generation Context-Aware Services. *IEEE Network*, **22**(4):34-42. http://dx.doi.org/10.1109/MNET.2008.4579769

COSTA, C.A.; YAMIN, A.C.; GEYER, C.F.R. 2008. Toward a General Software Infrastructure for Ubiquitous Computing. *IEEE Pervasive Computing*, **7**(1):64-73. http://dx.doi.org/10.1109/MPRV.2008.21

EJIGU, D.; SCUTURICI, M.; BRUNIE, L. 2008. Hybrid Approach to Collaborative Context-Aware Service Platform for Pervasive Computing. *Journal of Computers*, **3**(1):40-50. http://dx.doi.org/10.4304/jcp.3.1.40-50

FERRY, N.; HOURDIN, V.; LAVIROTTE, S.; REY, G.; RIVEILL, M.; TIGLI, J.-Y. 2013. WComp, Middleware for Ubiquitous Computing and System Focused Adaptation. *In:* N. FERRY; V. HOURDIN; S. LAVIROTTE; G. REY; M. RIVEILL; J.-Y. TIGLI. *Computer Science and Ambient Intelligence*. Hoboken, John Wiley Sons, p. 89-120. http://dx.doi.org/10.1002/9781118580974.ch6

ISTA - SEED TESTING INTERNATIONAL. 2012. *ISTA News Bulletin.* N. 144, p. 111.

KAKOUSIS, K.; PASPALLIS, N.; PAPADOPOULOS, G. 2010. A survey of software adaptation in mobile and ubiquitous computing. *Enterprise Information Systems*, **4**(4):355-389. http://dx.doi.org/10.1080/17517575.2010.509814

KNAPPMEYER, M.; KIANI, S.; REETZ, E.; BAKER, N.; TONJES, R. 2013. Survey of Context Provisioning Middleware. *Communications Surveys Tutorials, IEEE*, **15**(3):1492-1519. http://dx.doi.org/10.1109/SURV.2013.010413.00207

LOPES, J.; SOUZA, R.; COSTA, C.; BARBOSA, J.; GUSMÃO, M.; YAMIN, A.; GEYER, C. 2012. A Model for Context Awareness in Ubicomp. *In:* BRAZILIAN SYMPOSIUM ON MULTIMEDIA AND THE WEB, 1, São Paulo, 2012. *Anais…* New York, **1:**161-168. http://doi.acm.org/10.1145/2382636.2382672

SILVA T.; CELES C.; MOTA V.; LOUREIRO A. 2012. A Picture of Present Ubicomp Research Exploring Publications from Important Events in the Field. *Journal of Applied Computing Research*, **2**(1):32-49. http://dx.doi.org/10.4013/jacr.2012.21.04

WEISER, M. 1991. The Computer for the 21st Century. *Scientific American*, **3**(265):94-104.

YOON, C.; KIM, S. 2007.Convenience and TAM in a Ubiquitous Computing Environment: The Case of Wireless LAN. *Electronic Commerce Research and Applications*, **6**(1):102-112. http://dx.doi.org/10.1016/j.elerap.2006.06.009

Submitted on October 21, 2013
Accepted on January 25, 2014