A Distributed Architecture for Dynamic Context Awareness in UbiComp

João Lopes, Rodrigo Souza, Cláudio Geyer
PPGC/INF/Federal University of Rio Grande do Sul
Porto Alegre-RS, Brazil
{jlblopes,rssouza,geyer}@inf.ufrgs.br

Adenauer Yamin
PPGC/CDTec/Federal University of Pelotas
Pelotas-RS, Brazil
adenauer@inf.ufpel.edu.br

Abstract

Ubiquitous Computing (UbiComp) environments are characterized by high distribution, heterogeneity and dynamism. In these environments, applications must be aware of their contexts and adapt to changes in them. Considering these characteristics of ubiquitous environments, this paper presents an architecture for context awareness, called DynamiCA (Dynamic Context Awareness), which includes elements to support contextual data collection, environment actuation, and processing of contextual information. We consider that the main contribution of this work is the proposal of an architecture that supports acquisition, storage, processing, and dissemination of context, in a distributed way, independently of the application, in autonomic and rule-based perspective. To assess the functionality of the proposed architecture, we present a case study, highlighting prototyping performed, technologies employed, and tests realized.

1. Introduction

The computational infrastructure for UbiComp tends to be highly distributed, heterogeneous, and dynamic. This is because of the contextual elements, devices, and software components that often change their status in the ubiquitous environment. Therefore, one of the research challenges in this area concerns the ability of software development resources to explore the context awareness for the adaptation process of applications [9].

In computational environments for ubiquitous applications, contextual information is usually obtained from multiple sensors. Consequently, the software components should be aware of their contexts of interest and, when appropriate, automatically adapt to their changes, characterizing by that context awareness [4].

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 on the environment, and (iii) context dissemination to interested consumers in a distributed and timely way [3] [2].

Considering the integration with EXEHDA middleware [8] this paper presents the current state of DynamiCA architecture. In this architecture, the context data obtained from various sources can be dynamically processed and compounds in contextual information at a higher level, which can be used in the adaptation process of ubiquitous applications at runtime.

The paper is organized as follows. Section 2 describes the software architecture. Section 3 highlights the prototypes and tests. Section 4 summarizes related work. Finally, section 5 presents the concluding remarks.

2. Architectural Model

The software architecture consists of modules distributed between the two types of servers: Context and Border. 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 Context Server is responsible for processing the contextual information. In turn, the Border Server is responsible for interaction with the environment, through sensors and actuators. Figure 1 shows an overview of the DynamiCA software architecture, highlighting the Context Server.

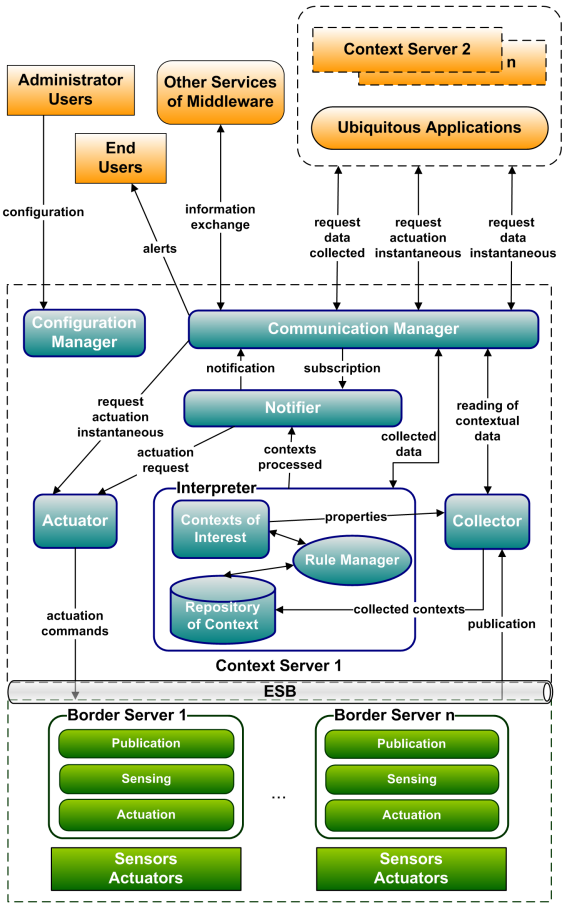

Figure 1. Context Server

2.1. Context Server

The Context Server is organized into six autonomous modules, described following, that interoperate in the provision of the necessary functionalities for context awareness services.

Collector Module provides support for the capture of contextual information, collected by Border Servers, considering logic and/or hardware sensors. In turn, **Actuator** Module triggers the ubiquitous environment actions that change the state of the environment.

The **Interpreter** processes contextual information based on Contexts of Interest of applications. This module maintains a Repository of Context, which employs a relational model for the context representation. In this repository are stored the information obtained by the Collector, providing a historical view of contexts. These data are used by the Rule Manager Component, which triggers the appropriate actions, depending on the context status. The nature of the rules (logical, numeric, temporal...) is a consequence of the application domain managed by DynamiCA.

Notifier Module receives, through the Communication Manager, subscriptions from all services and/or applications that requiring notifications about the context state. The Notifier also receives all decisions of actuation, resulting from the autonomic treatment of context rules.

The **Communication Manager** provides the dissemination of context information to other middleware services, as well as it can send messages to users. In turn, **Configuration Manager** Module allows the management of Context Server settings, including specifications of sensors and actuators, as well as information of equipment which the context is being collected.

2.2. 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 2.

Sensing Module provides handling of a sensor networks, enabling the individualization of processing by sensor. It covers aspects from physical management (interfaces, reading frequency) until computational normalization (validation, translation) of the collected values. Also provides features for publication the information collected from sensor networks in Context Server. This module consists of six components described following.

Operating Parameters of Sensors Component specifies the driver to be used for reading different sensors, as well as the schedule to data acquisition. Based on this information, the Scheduler Component triggers readings, considering the user's application interest.

The Validator assesses whether a particular data collection should be published or not, using criteria specified by user-defined rules. Translator Component performs the adequacy of the collected data to the nature of the user application, also by a rule.

Instantaneous Reading Component enables reading of a sensor, considering application's demand, at any time. It employs an ESB (Enterprise Service Bus) for receiving asynchronous requests and, considering the sensor ID, triggers the corresponding driver. Still, the Local Rules Handler processes contingency rules intended to prevent the devices involved reach critical states.

Publication Module is responsible for coordinating the main data flow between Border Servers and Context Server, promoting the publication of all collected data and ensuring a Local Persistence in the periods that the publication be frustrated. This module is composed by Publisher Component, which interoperates with Collector Module of the Context Server, performing submissions of collected data.

Actuation Module is responsible for managing the actuators. This module is composed by Instantaneous Actuation Component, which receives asynchronous commands at any time, via an ESB bus. These commands are originating from the Context Server, as a result of the execution of a rule, as well as from a user application, and from Local Rules Handler Component. Also is part of this module the Supervisor Component, which active the driver related to the actuator that is being managed. The actuator's drivers has 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.

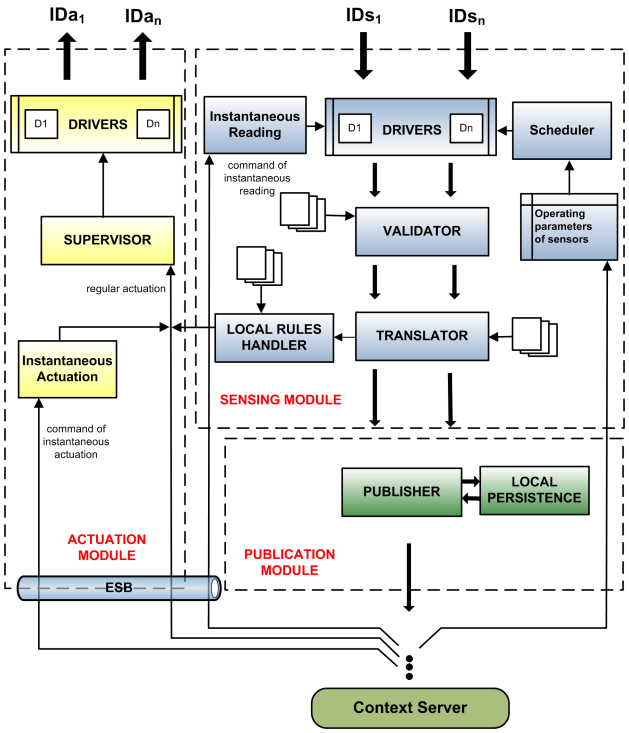

Figure 2. Border Server

3. Prototyping and Testing

The DynamiCA architecture is not specific to a particular problem domain, but is designed to be comprehensive, aiming to meet different domains. In this sense, despite of the fact that the applications described in this section are related to project AMPLUS[1] in the area of agriculture, the DynamiCA enables support in other application domains.

The prototypes and tests evaluated some characteristics of the software architecture, and the technologies employed, covering aspects related to data collection and actuation on the environment, and composition of dynamic contexts.

The code of Context and Border servers is written in Python language, using XML-RPC (*Extensible Markup Language - Remote Procedure Call*)[2] to implement the ESB used for interoperability. In the Repository of Context is employed PostgreSQL to implement the databases. The sensors and actuators communicate through 1-Wire protocol[3].

A Border Server manages a sensor and actuator networks, which communicate using the 1-Wire protocol. These servers collect the data produced by sensors associated with the different contexts, as well as trigger commands on existing actuator in the environment. For the collection and actuation process we used, respectively, digital temperature sensors and an electronic key that controls a warning light, both based on 1-Wire technology. We configured the Publication Module of the Border Server to publish the temperatures every 30 minutes in the Context Server.

For testing data collection, actuation, and composition we considered the following initial profile of the computational environment:

- Environment: Didactic Laboratory of Seed Analysis (LDAS)
- Collector: LDAS Border Server
- Application: Control of Physical Environment Condition of LDAS
- Component: Temperature Control in the Room of Seed Germinators
- Context of Interest: temperature
- States of Context: temperature values collected
- Event: temperature outside the boundary ($\geq 27^{o}$ C or $\leq 15^{o}$ C)
- Rule: Activate Warning Light (triggers the software component that activates a warning light in the room of the responsible for LDAS)

To test the composition of dynamic contexts, the context of interest has been modified to include the factors humidity and temperature of the external environment. This last factor is published by a Border Server, managed by the Context Server located at Embrapa-Lowlands cell, specialized in Agroclimatological Station and located at Capão do Leão Campus (campus of Federal University of Pelotas, where the LDAS is situated). This setting

1 http://amplus.ufpel.edu.br

2 http://www.xmlrpc.com

3 http://ubiq.inf.ufpel.edu.br/1-wire/

characterizes a collection of context data in a distributed way, in different cells of the ubiquitous environment.

We validate the modifying of processing rule, at runtime, using transactions. When a state of context occurred that triggers an event related to the rule "Activate Warning Light", outside the operation hours of the laboratory, we began to be used the rule "Send E-mail", which triggered the software component to send an email to the person responsible for the laboratory, warning the occurrence of a temperature outside the established boundary.

In this scenario, the Border Server LDAS published the context data (temperature and humidity) collected in the laboratory room, and the Context Server added the external temperature published by the Border Server of Agroclimatological Station and made the composition, considering the states of context, events and related rules.

4. Related Work

The study of related work (CARE [1], CoCA [6], HiCon [5], WComp [7]) has been done considering the main design assumptions of DynamiCA: (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 DynamiCA is structured in a distributed way, at all stages of handling context information, from collection to the actuation.

The DynamiCA can manage sensor and actuator networks; such feature is found in part in projects CoCA and HiCon, which have support to sensor networks. The project WComp allows actuation on the environment, however, does not support actuator networks.

With the exception of the CARE project, the others allow the use of specific mechanisms for acquisition, adopting a strategy of separation between the collection and use of context. Besides contemplating this aspect, the DynamiCA 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 been in operation.

In most projects, the handling of rules is restricted to a few steps of the context processing. The DynamiCA 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.

5. Concluding Remarks

The main contribution of DynamiCA corresponds to the support for context awareness, implemented through an architecture that enables the acquisition, storage, processing, and dissemination of context information, in a distributed way, independently of the application, in autonomic and rule-based perspective.

Among others, the following aspects should be explored in future works: (i) revise and expand the aspects considered in the target applications of the proposal; (ii) specify the API and protocols between the application layer and middleware, and (iii) continuing the specification of the architecture modules.

References

[1] A. Agostini, C. Bettini, and D. Riboni. Hybrid reasoning in the care middleware for context awareness. *Int. J. Web Eng. Technol.*, 5(1):3–23, May 2009.

[2] P. Bellavista, A. Corradi, M. Fanelli, and L. Foschini. A survey of context data distribution for mobile ubiquitous systems. *ACM Comput. Surv.*, 44(4):24:1–24:45, Sept. 2012.

[3] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, and D. Riboni. A survey of context modelling and reasoning techniques. *Pervasive Mob. Comput.*, 6(2):161–180, Apr. 2010.

[4] R. Caceres and A. Friday. Ubicomp systems at 20: Progress, opportunities, and challenges. *Pervasive Computing, IEEE*, 11(1):14–21, 2012.

[5] K. Cho, I. Hwang, S. Kang, B. Kim, J. Lee, S. Lee, S. Park, J. Song, and Y. Rhee. Hicon: a hierarchical context monitoring and composition framework for next-generation context-aware services. *Network, IEEE*, 22(4):34 –42, july-aug. 2008.

[6] D. Ejigu, M. Scuturici, and L. Brunie. Hybrid approch to collaborative context-aware service plataform for pervasive computing. *Journal of Computers*, 3:40–50, 2008.

[7] N. Ferry, V. Hourdin, S. Lavirotte, G. Rey, M. Riveill, and J.-Y. Tigli. Wcomp, a middleware for ubiquitous computing. In E. Babkin, editor, *Ubiquitous Computing*, volume 1, chapter 8, pages 171–176. InTech, 2011.

[8] J. L. Lopes, R. S. Souza, M. Z. Gusmao, C. A. Costa, J. V. Barbosa, A. C. Yamin, and C. R. Geyer. A model for context awareness in ubicomp. In *Proceedings of the 18th Brazilian symposium on Multimedia and the web*, WebMedia '12, pages 161–168, New York, NY, USA, 2012. ACM.

[9] T. Silva, C. Celes, V. Mota, and A. Loureiro. Overview of ubicomp research based on scientic publications. in:. In *Simpósio Brasileiro de Computação Ubíqua e Pervasiva*, Curitiba, PR, 2012. SBC.