Using Discovery and Monitoring Services to Support Context-Aware Remote Assisted Living Applications

André Luiz B. Rodrigues¹, Izabela C. Gomes², Leila N. Bezerra¹, Alexandre Sztajnberg¹², Sérgio T. Carvalho³⁴, Alessandro Copetti³, Orlando Loques³

¹PEL and DICC12 / Universidade do Estado do Rio de Janeiro (UERJ) - Rio de Janeiro, Brazil
³Instituto de Computação / Universidade Federal Fluminense (UFF) - Niterói, Brazil
⁴Instituto de Informática / Universidade Federal de Goiás (UFG) - Goiânia, Brazil

{rblandre, izabelacg, leila}@ime.uerj.br, alexszt@uerj.br
{scarvalho, acopetti, loques}@ic.uff.br

Abstract

Ubiquitous and pervasive applications are aware of the context of the used resources. This class of application can benefit from mechanisms to discover resources (devices and sensors) that meet their requirements and mechanisms to monitor the state and provide access to the functionalities of these resources. We present an architecture that includes two essential services to compose the supporting infrastructure required by the mentioned applications: a Context Service, that provides access to context information, and a Discovery Service. Resource Agents, which encapsulate and provide access to the actual resources and sensors, are also included in the proposal. A reference implementation, based on Web Services technology, was used to develop a remote assisted living application, which relies on ambient sensors, placed in each room, to capture context information and on a combo medical appliance to perform blood pressure measurements on the patient, according to a care plan prescribed by a doctor. The collected set of context data may be transmitted to a monitoring center and also interpreted locally using medical knowledge; the identification of a patient's abnormal condition can activate local actions or send an emergency message to a monitoring center. The local infrastructure can help a doctor to monitor and assist the patient while he or she is performing daily activities. It may also help reducing hospitalization time and improving the patient's treatment compliance and quality of life.

1. Introduction

Context-aware applications are constrained and bound by the state of the used components and resources, regarding their availability and quality. For instance, considering context-aware remote assisted living applications, the environment where they are usually deployed can often change: sensors or new medical equipments can be added, removed or stopped. As a consequence, these applications need to adapt due (i) to resource and context variation, (ii) to new features introduced in the application and (iii) to the operation quality expected by the user application. An adaptation is often triggered by a set of rules or policies that take into consideration the current state of the application, its components and used resources, as well as domain-related information such as, the current activity and location of the patient, health status, whether or not the patient lives alone, or a new medical routine prescribed by a doctor. Such class of applications can benefit from mechanisms to discover resources that meet their non-functional requirements and mechanisms to monitor the quality of those resources.

The activity of discovering and monitoring resources is recurrent in ubiquitous systems, which have diverse applications, such as smart spaces, home automation or e-health. It is desirable that these activities can be treated with high-level abstractions, and managed in a uniform way.

To provide such a high-level discovery and monitoring capabilities the infrastructure has to offer basic services and enforce some discipline. For
instance, components and resources to be discovered, used and monitored have to be described and then published (or registered) in some directory service. Without these steps clients would not be able to find the services nor confirm if a service really offers what is needed.

The description style of components and resources should be decoupled from specific implementations and should facilitate the integration of new elements. Moreover, interaction among resources and the infrastructure should also be independent from specific communication mechanisms so they can be integrated and reused more easily in other domains.

The first part of the article presents a set of essential services that are included in the management infrastructure of the applications: a Context Service that provides access to context information; and a Discovery Service, which allows the dynamic discovery of resources, considering context constraints to be satisfied. These services rely on agents, which monitor the actual resources. The software architecture of these services was designed as a framework of objects, which are implemented as Web Services and follow simple design patterns. Examples from the pervasive computing domain demonstrate the facilities of the proposed services.

In the second part of this article we sketch a context-aware remote assisted living application, which employs these services. In this application we use sensors in each room to capture ambient context information, and a combo wireless medical appliance to collect the patient’s blood-pressure measurements according to a care plan prescribed by a medical doctor; the patient’s current activity is also inferred using collected information. This data set is used to detect and report abnormal situations (sending alert or emergency messages to a monitoring center) and can help a doctor to monitor and assist the patient while he or she is performing daily activities. It may also allow reducing hospitalization time and help increasing the quality of life, mainly for elderly people.

2. Discovery and context services

The proposal of this work considers that when deployed a context-aware application needs to find resources of a given kind to begin execution and that these resources have to comply with certain context restrictions (specific characteristics or operation properties). Once a resource is found and selected, the application obtains a reference to use it and monitor its state. An application can need to monitor several resources to reason on the current context (i.e., the set of states of all monitored resources) and, as a consequence, either decide to maintain the current configuration or to trigger adaptation procedures. This can happen frequently in some environments, e.g., when a patient moves from one room to another or begins a new activity. An adaptation can imply discovering better resources of an already used kind or discovering a new kind of resource.

Given the mentioned requirements, a model of services is proposed, comprising the following elements:

**Resource Agent (RAgent).** A Resource Agent acts as a proxy that makes context information available and responds to monitoring queries on behalf of a given resource, hiding low-level details used in the sensing and acquisition of raw data. In this regard a uniform interface has to be provided, and the application does not have to be aware of the specific sensing mechanisms.

**A resource type classifies the resources of a given kind. Each resource type is associated to an XML description, which specifies its properties, and is registered in the Registration and Directory Service. Each resource type has the support of a corresponding specific RAgent. As mentioned this RAgent has to provide the context information of the resource it represents based on the XML description, regardless of how it is actually measured or sensed.**

**Registration and Directory Service (Reg&DirServ).** The Registration and Directory Service is the prime repository for all resource discovery and monitoring elements. At startup each RAgent has to register itself in the Reg&DirServ to inform the class of resource it belongs to, its properties, and if each property is static or dynamic.

**Context Service (ContextServ).** The Context Service is responsible for making available context information and hiding the communication details from the RAgents. It provides a high-level access interface for information requests. A client application can customize which data is required and how it shall be collected.

**Discovery Service (DiscServ).** The Discovery Service provides an interface that can be used to identify among registered resources a list of resources of a given kind that complies with a set of context constraints. The returned list can be further filtered or ranked so that the best possible resource is identified.

2.1. Architecture

The overall architecture of the framework, composed by the services and their relations, is
presented in Figure 1. The Reg&DirServ receives registration requests from all RAgents. When performing a discovery the DiscServ goes to the Reg&DirServ and gets a list of resource locations. If one or more attributes have dynamic values, the DiscServ has to ask the ContextServ for recently monitored information. The communication between the RAegnts and the ContextServ can be configured to work proactively or reactively, i.e., an RAgent can notify the ContextServ every time a monitored attribute has its value changed, or respond to on-demand queries from the ContextServ.

Figure 1. Framework components

The DiscServ and the ContextServ interfaces are provided for the client. One can query the DiscServ to find a resource of a given type and a given set of attribute values or range of values. The DiscServ returns a list of references of resources that satisfy all constraints. After setting the initial configuration of resources, the context-aware application can monitor the context using the ContextServ. The interaction between the client and the ContextServ can also be set to push or pull styles. A query submitted to the ContextServ can be customized so that a set of RAegnts is monitored. The resulting response is a composition containing information of all RAegnts in the set.

2.2. Resource representation

For consistency reasons every resource type description (that works as a template and as a contract) has to be initially registered in the Reg&DirServ. After that the respective RAegnts can register themselves in the Reg&DirServ and once registered the RAgent is ready to run. Figure 2 presents the description of a BloodPressure resource. First, the type is declared (line 2) and detailed description can be provided (line 3). After that a list of attributes can be declared (lines 4 to 9). Each attribute has a Name, a Type, and a Unit. Every RAgent registering as a BloodPressure resource in the Reg&DirServ has to monitor and provide these attributes.

2.3. Reference Implementation

A reference implementation of the proposed services, named the Context Discovery Resource Framework (CDRF), was developed as a framework of components and classes using Java V.6. The principles in this implementation are (i) minimum dependency among the services; and (ii) minimum dependency on communication protocol.

The abstract classes and interfaces of the framework, which contain the standard behavior of the services, were specialized as Web Services. In this regard we used the XML Web Services API, JAX-WS 2.0. This API makes it possible to implement communication among our services and the RAegnts through RPC using SOAP 1.0 messages.

The services and the RAegnts were deployed as stand-alone daemons, each of them running as Web Services over an independent Web server. We used the embedded HTTP server provided by the standard JDK 6.0, which was prepared to forward the requests to our server classes with no need to install and configure complete Java EE application servers.

3. Remote assisted living application

We are evaluating the reference implementation of the services employing it as the infrastructure to run a context-aware Remote Assisted Living application, which caters for monitoring elderly patients in their homes.

A typical setting involves sensors that collect data from the patient and send this data to a monitoring center (a clinic, a doctor, a hospital). In our approach, we incorporate local intelligence at home: ambient sensors in the house and sensors worn by the patient constantly generate data, which is collected by a local computer system that interprets them using medical knowledge. The identification of a patient's abnormal condition can activate a local device (e.g., turning on an air conditioning appliance), start an interaction with the person (e.g., through a TV screen) or send an emergency message to a monitoring center.

```
<resource>
  <type>BloodPressure</type>
  <description>Blood Pressure Resource Agent</description>
  <attributes>
    <attrib name="Serial" type="string"/>
    <attrib name="Model" type="string"/>
    <attrib name="systolic" type="integer" units="mmHg"/>
    <attrib name="diastolic" type="integer" units="mmHg"/>
  </attributes>
</resource>
```

Figure 2. BloodPressure resource description
3.1. Home Health System

The HHS is a software system built to collect, store, analyze and distribute context data. In our approach, context data is comprised by the ambient sensor measurements, the patient’s current activity and her/his location in the house, and the physiological/medical measurements. The HHS evaluates the collected context data to infer the patient’s health status. All context data and the inferred health status are stored in a local database to maintain an individualized history of the patient. The history information is also sent to a monitoring center and is made available to the doctor and other involved health professionals and some of the patient’s relatives.

![Home Health System Organization Diagram](image)

**Figure 3. Home Health System organization**

The software system core is the Care Plan, which is composed by medical prescriptions and routines directed to the patient. Figure 3 presents the overall organization of the system modules:

- Monitoring: discovers and monitors sensors and appliances to collect context data;
- Analysis: applies a series of rules and uses a fuzzy model to infer the patient’s status, severity trends or critical situations;
- Persistence: uses a relational database to store collected context data and inferred status;
- Scheduler: can be programmed to send alert messages, transmit context data to the monitoring center and to trigger pre-configured actions at specific times;
- Distribution: responsible for forwarding data to a monitoring center and for sending notifications or detailed data to the doctor or health professionals;
- Interaction: handles the presentation of the monitored data, providing appropriate visions to each actor of the system, i.e., the patient, family members and authorized health professionals.

More details on the HHS and on the intelligent analysis module can be found at [3].

3.2. The Scenario

The scenario we are working on is depicted in Figure 4. Each room has a set of ambient sensors (temperature, light and humidity). The heating and cooling system of the house and the Digital TVs (DiTV) that are able to receive and display messages are also part of the application. The patient will wear a location tag and a motion sensor (an accelerometer).

![Remote Assisted Living Ambient Diagram](image)

**Figure 4. Remote assisted living ambient**

The patient will also have to put on a medical appliance when the care plan says it is time to do so. All devices communicate with a computing node using a mix of wireless and wired network. A Home Health System (HHS) unit runs on the computing node a software system comprised by a database, a communication module, an intelligent scheduler and a specialized browser.

Each sensor or device will be represented by an RAgent. The software running on the HHS relays on the DiscServ and ContextServ to collect context information pushing or pulling data to/from the RAgents.

3.3 Deployment

The daily use of the proposed system has to consider that the patient or the accompanying care providers are not computer experts. So, the use of the system has to be preceded by an installation and deployment phase, which comprises the following steps:
Basic deployment. The ambient sensors are installed, adjusted and calibrated. The HHS is loaded in the computing node and appropriate security (passwords, keys or digital certificates) is configured;

HHS configuration. Basic information has to be initially configured: the house map (in our prototype, the name of each room) to allow locating the patient, and the URI references for the supporting services;

Communication configuration. Communication with the central system, doctors (or care providers) cell phone are set and configured;

Sensor description and registration. Each sensor is described, according to its RA isent class and registered in the Reg&DirServ;

Medical appliances selection. A nurse or a care provider has to retrieve the initial Care Plan and select the appropriate set of medical appliances;

Medical appliances registration. After selecting the device, the corresponding RA isent description is customized with the help of the HHS GUI and the corresponding RA isent is also registered in the Reg&DirServ.

Once the previous steps are completed the system can perform a sensor survey. The sensor survey consists in consulting the DiscServ to find all available ambient sensor RA isents registered in the Reg&DirServ for each room. This information confirms which sensors are expected to provide ambient data for each room during normal operation, and allows detecting malfunctioning of a sensor afterwards.

3.4 Operation

At this point the system is configured with all necessary information and can begin operation. When it comes the time to perform a physiological measurement, the patient (or the care provider) has to correctly place the device and make the measurement. This step could be unnecessary if the medical device could be worn all the time or if it could be programmed to automatically initiate a measurement according to the Care Plan schedule. The RA isent representing the medical device notifies the HHS of the new measurement.

The obtained measurement is parsed, the relevant values are stored in the database, and then the ContextServ is consulted to get the context from the UserLocation service RA isent in order to identify in which room the patient is currently located. With the user location the HHS engine can then get the current values of all ambient sensors (Figure 5).

3.5. Prototype

Using the reference implementation, we developed a first prototype of the remote assisted living application. We are using a wireless device, named WristClinic [5], which is able to measure blood pressure, heart rate, SpO2, body temperature and breathing rate. In this prototype we employ this device to get blood pressure measurements and a PDA to manually inform the patient’s activity. Based on this information a fuzzy rule system infers the patient’s health status [3]. All collected data is stored in a PostgreSQL database. Figure 5 presents a sample of the Graphical User Interface. Currently we are making some extensions in our prototype:

- Ambient sensors RA isents are being integrated to the system. We are currently using temperature loggers in our tests. The inclusion of procedures to activate or reconfigure ambient devices is also under development. Some experiments were performed with X10 devices (www.x10.com) to switch on-off and dim the room lights according to the patient’s current location;
We are also considering power saving concerns based on the patient’s current location and care plan. In order to prolong battery life, only the sensors located in the same room will be kept fully active;

The distribution module was implemented and is being integrated with the reminder system. Depending on the message or warning to be sent, and on the severity of the information, the DiTV or mobile phones can be used, in addition to sending context information and warnings to the monitoring center.

4. Concluding remarks

We proposed an architecture to support context-aware applications, which includes two key-services: (i) the ContextServ, used to collect information of the resources of the application and its operating environment, and (ii) the DiscServ, which allows discovering resources dynamically considering context restrictions. The integration of these services in ubiquitous and pervasive applications is based on meta-level descriptions of resource classes, which are registered in the Reg&DirServ, and relies on a neutral interaction protocol.

We developed a reference implementation for the services, which employs proved software engineering techniques including design patterns, in addition to abstract classes and XML descriptions. In particular, devices and sensor modules are deployed as Web Services and interact through SOAP.

Remote assisted living applications are not a novelty, e.g., [1, 2, 4]. However, the integration of medical measurements and other context information, obtained from ambient sensors and wearable devices, provided by our approach adds a further step in this area.

The implementation of the remote assisted living application is currently in progress, though its relevant modules are already implemented and tested. The development of the aspects that depend on specific medical expertise is supported by medical collaborators, from the UERJ (University Hospital) and UFF (Biomedical Institute). Some further improvements we are planning with their help are worth to mention:

- Refinement of the rules to infer health status, mainly those related to the precise classification of critical situations and emergency alerts;
- Self-adaptation of the Care Plan. A new prescription could be deployed by the doctor, or the care plan management engine could recommend adjustments (to the current plan) based on an inferred trend.

We are planning to start soon an evaluation of the application with real patients. Besides verifying the benefits accrued from the use of the system, we want to evaluate the effects of the daily interaction of the patient with the system and also receive the critics from the health providers behind it.

Acknowledgments. The authors would like to thank FAPERJ and CNPq for their support. The authors also would like to acknowledge Thais Barbosa and Antonio da Nóbrega, from UFF, and Mario Fritsch and Sérgio Freire, from UERJ, for their support on medical issues. We also thank our students David Ferreira, Evandro Macedo and Rodrigo Santos, from UERJ, and Matheus Erthal and Douglas Mareli from UFF.

References