PICO

DELIVERABLE 2.3

Prototype

Questions about this document or the features it describes should be directed to:

CEFRIEL
Via Fucini, 2
20133 Milano (MI)
Italy

Politecnico di Torino
Corso Duca degli Abruzzi, 24
10129 Torino (To)
Italy

Abstract

The PICO project concentrates on application streaming and context awareness as typical techniques to build distributed applications in the domain of emergency situations (described in D1_1). This document describes an approach to implement the context-aware platform based on service adaptation.

CHANGE LOG

Version	Date	Description
1.0	18/11/2009	This is the first draft of the Deliverable 2.3
2.0		Final version

TABLE OF CONTENTS

Contents

Introduction

In this document we will present a approach for gathering, exchanging and providing user context, in order to handle all types of user information and define proper interfaces for interactions over mobile networks. This architecture is essentially defined by: Context Manager Client (context acquisition and dissemination) and Context Manager Server (manipulation, representation, recognising and reasoning about context and situations).

1. Architecture

A technological approach to be used for gathering user context is based on either: device-based (some application in the end-user device that captures information related to the user), or network-based (some system in the network that holds some information about the user). Since some information resides on the end-user device only, and other is only available in the network, both approaches are complementary.

An architecture for gathering, exchanging and providing user-related information (context), has to be able to handle all types of user information and define lightweight XML/HTTP interfaces for interactions over mobile networks. This architecture is essentially defined by two main parts:

- Context Manager Client is in charge of context acquisition and dissemination.
- Context Manager Server is in charge of manipulation, representation, recognising and reasoning about context and situations.

Taking into consideration the full sense-decide-actuate cycle of context-awareness then two other subsystems can be identified:

- Sensor and sensor network subsystem (logical and physical sensors).
- Service Adaptation Component: Once the system recognizes that some adaption is needed, it adapts the service according to reasoning techniques applied on the context information, after this, the service is automatically triggered and delivered.

When the user requests a service in an emergency situation, the Context Manager Client component captures the user's context and sent it to the Context Manager Server. Service Adaptation component verifies the compatibility between the user's context and the service context. If the two contexts are not compatible, Services Adaptation component triggers an adaptation process. The service is updated by this component after each adaptation process.

1.1 Context Manager Server

The ContextManagerServer collects all the context data coming from the end-user device, in order to extract the information contained in the contextML message sent from the *ContextManagerClient*.

1.1.1 Local Context Data Request

The *ContextManagerServer* (Figure 1) invokes the *ContextMLParser* component with the purpose of providing the info requested from the *AppDownloadManager* component. The context interfaces provided through the **ContextManagerServer** have been designed according to:

- **User info:** The information related to user name and user ID (Fiscal Code or so – unique code) will be provide from two interfaces defined as:
  - *Public String getCtxUserName()*
  - *Public String getCtxUserID()*

- **Resource info:** The interfaces related to this context category will be represented by relevant device info:
  - *Public String getTypeOfDevice()*
  - *Public int getNetworkState()*
  - *Public String getJavaVersion()*
  - *Public String getOSType()*
  - *Public int getOSVersion()*
  - *Public int getScreenHeight()*
  - *Public int getScreenWidth()*
  - *Public int getAvailableMemory()*

- **Location info:** The interfaces for this category were defined as:
  - *Public String getLatitude()*
  - *Public String getLongitude()*
  - *Public int getTemperature()*
  - *Public int getLighting()*
  - *Public int getTypeOfArea()*

- **Temporal info:** This context category provides info related to:
  - *Public int getHourOfDay()*
  - *Public String getDayOfWeek()*
  - *Public Boolean isFestivity()*

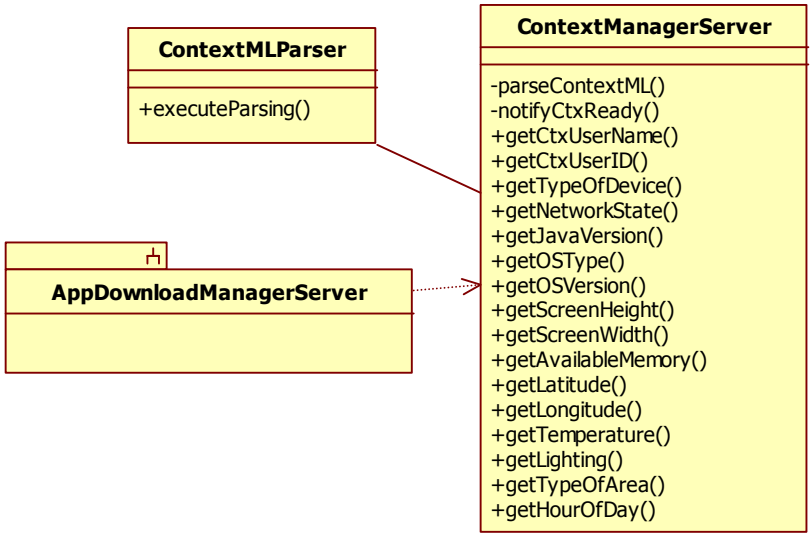

Figure 1. Class diagram – server side

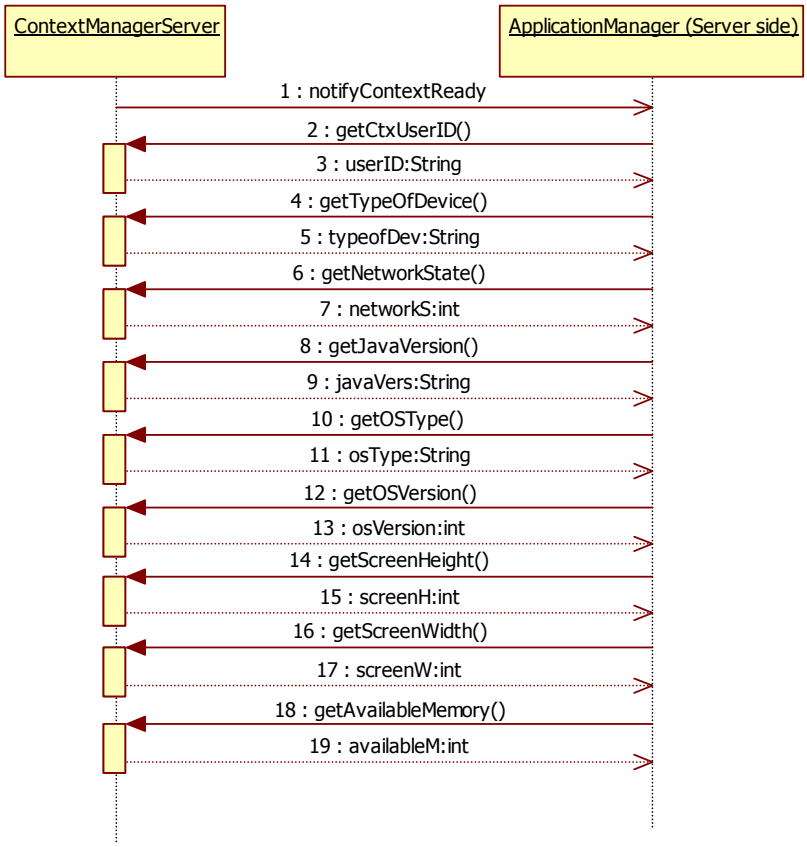

Figure 2. Sequence diagram – server side

1.1.2 Remote Context Data Request

The *ContextManagerServer* invokes the *ContextMLParser* component with the purpose of providing the info requested from a remote component. The context interface provided through the ContextManagerServer has been designed according to:

- **User context:** The information is retrieved from the ContextMLParser component and provided to the requestor. This interface receives as inputs username (who retrieves the information) and the list of scopes. If the whole functionality can be performed successfully, the ContextManagerServer returns the user information. If the names of the scopes are wrong, the ContextManagerServer returns an error message.
  The remote context data request is performed by sending an HTTP GET request to the target URL containing the username and scopeList as URL parameters. For example, "jack", asks for "cell" scope to the ContextManagerServer.

```
GET
http://myServer/.../getContext?entity=jack&scopeList=cell
HTTP/1.0
```

If there is not any error, the ContextManagerServer should answer the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<contextML>
    <ctxEls>
    <ctxEl>
    <contextProvider id="CMS" v="0.1"/>
    <entity id="jack" type="username"/>
    <scope>cell</scope>
    <timestamp>2009-11-04T09:00:00+01:00</timestamp>
    <expires>2009-11-15T09:01:00+01:00</expires>
    <dataPart>
          <par n="cgi">222-1-60923-11</par>
    </dataPart>
    </ctxEl>
    ...
    </ctxEls>
</contextML>
```

1.2 Context Manager Client

The *ContextManagerClient* is in charge of gathering context data by invoking interfaces provided through specific context managers: *AutenticationManager*, *ResourceManager*, *LocationManager* and *TemporalManager* (Figure 3). It gathers all the data related to the user, specially the data related to his actual context (connection available, device capabilities, etc). When the context info is collected, the *ContextManagerClient* is able to create the contextML message through the *ContextMLCreator*. Once the contextML message has been created, the *ContextManagerClient* is able to send it to the *ContextManagerServer* in order to be processed.

The Context Manager Client is composed of the sensors, and of the logic that collects all the available information and send them to the Context Manager Server. Sensors are pieces of code (Java APIs) able to get all the available information that allows describing the user context. They are written in a specific way to be recognized by the application, that contains all the logic needed to gather the specific context data.

The information collected by every sensor is stored in a XML document to be sent to the application every time someone asks for it. The Context Manager Client can control several sensors running at the same time. Every sensor is loaded at the beginning.

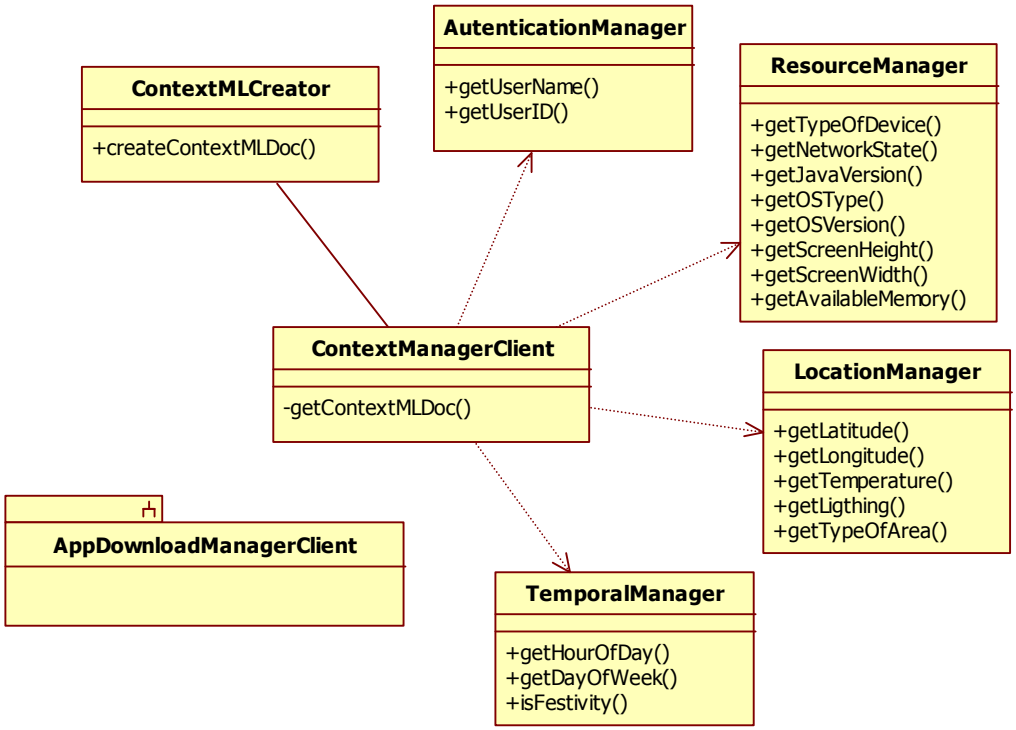

Figure 3. Class diagram – client side

Periodically the Context Manager Client logic asks each sensor for the data collected. Each one provides an XML description of the data. For every kind of different context information the sensor provide a *Context Element*, that describes a set of values related to a scope.

The Context Manager Client envelops all the Context Elements collected from all the sensors into a Document called *ContextML*. Once the users have logged in, their data are collected and periodically sent to the Context Manager Server as the body of an HTTP POST. Each scope has a scanning time of the information, dependent on how often the info changes.

For example, we can take a look at how the Context Manager Client retrieves user data and send it to the Context Manager Server (Figure 4):

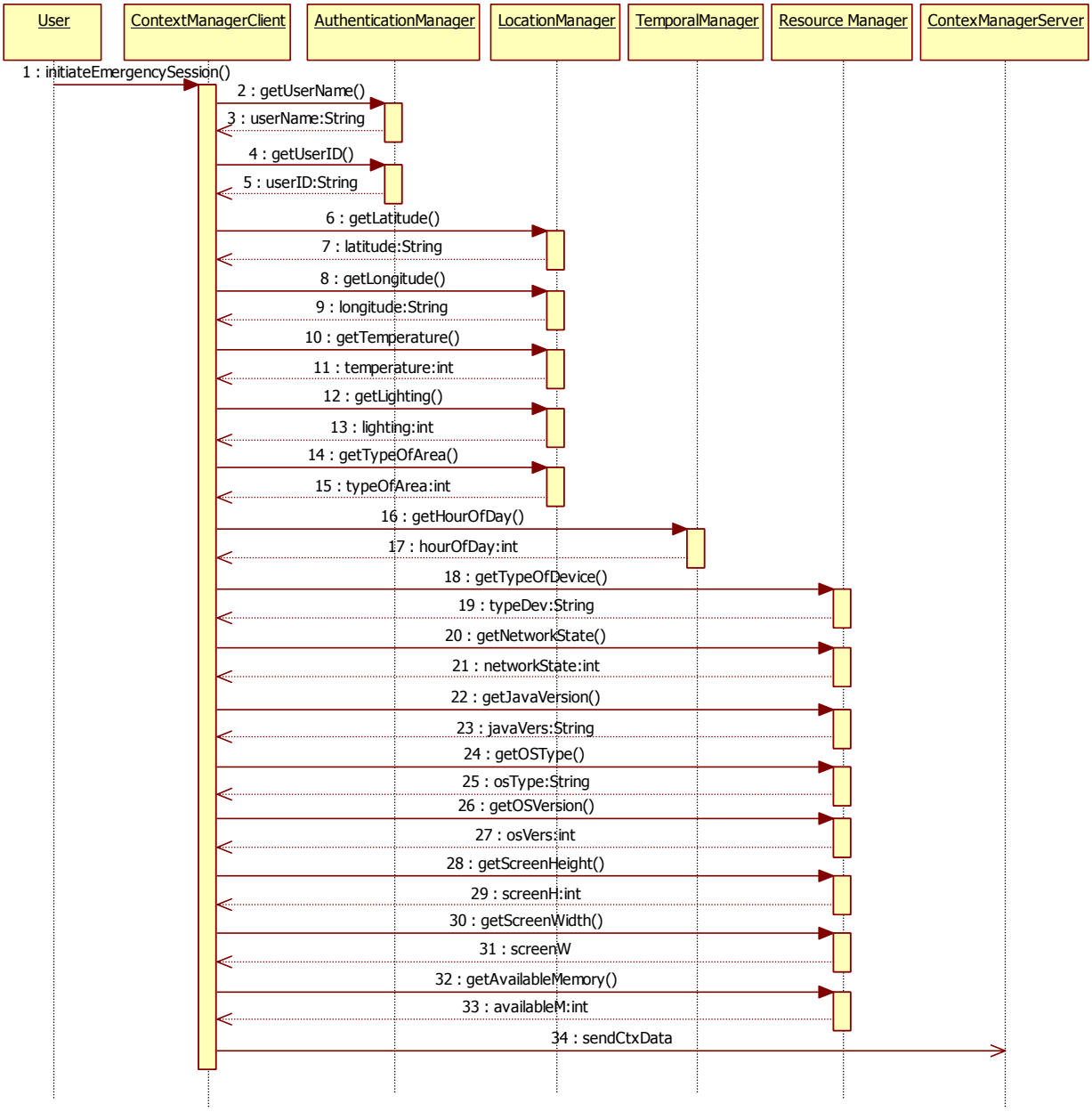

Figure 4. Sequence diagram – client side

ContextManagerClient invoke the method "sendCtxData" in order to asynchronously publish the information to the *ContextManagerServer*. If the data are correctly received, the *ContextManagerServer* will confirm the operation success. In case the data are wrong (configuration/translation process) an error occurs, then the *ContextManagerServer* will return an error message.

The aim of this application is to gather all the possible information from the user mobile terminal like Cell-ID, nearby Bluetooth, all possible device capabilities and so on, collect and send them to the Context Manager Server (Context Broker). Android is an excellent alternative to implement the main features of the ContextManagerClient, because it includes a powerful set of development tools.

These include a debugger, libraries, a handset emulator, documentation, sample code, and tutorials. It is currently supported by Linux, Windows XP or Vista. The officially supported integrated development environment (IDE) is Eclipse (3.2 or later) using the Android Development Tools (ADT) Plugin.

Appendix A Acronyms Table

XML	Extensible Markup Language
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
GUI	Graphic User Interface

Figures Index

Bibliography

[1] Situation Inference for Mobile Users: a Rule Based Approach, Laurent-Walter Goix Massimo Valla Laura Cerami Paolo Falcarin *Telecom Italia Lab, Italy Politecnico di Torino, Italy*

[2] Herlocker, J., Konstan, J. A.: Content-Independent Task-Focused Recommendation. In: IEEE Internet Computing, 5 (2001) 40-47

[3] van Setten, M.: Experiments with a recommendation technique that learns category interests. In: Proceedings of IADIS WWW/Internet, Lisabon, Portugal (2002) 722-725

[4] Jess:*"the Rule Engine for the Java Platform",http://herzberg.ca.sandia.gov/jess/*

[5] *http://en.wikipedia.org/wiki/Recommendation_system*

[6] www.wikipedia.com

[7] http://ieeexplore.ieee.org

[8] http://portal.acm.org

[9] IST *MobiLife Project* home page, *http://www.ist-mobilife.org*

[10] *http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html?page=1*

[11] *http://en.wikipedia.org/wiki/Widget_engine*