PICO

DELIVERABLE 2.2

Technologies

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 in detail the a technological approach to implement the service adaptation. We also introduce the most important technologies to support the prototype architecture.

CHANGE LOG

Version	Date	Description
1.0	18/11/2009	This is the first draft of the Deliverable 2.2
2.0		Final version

TABLE OF CONTENTS

Contents

Introduction

The aim of this document is to describe and analyze advanced technologies to adapt the service, in function of the attributes that identify the context [described in task 2.1]. In the document D2.1 we analyzed the language for the context description; and the description of a suitable adaptation process.
In this document we will present a technological approach for supporting context-aware functionalities (gathering user context and implementing adaptation), that is characterized by three logical levels. We will then introduce and describe these logical levels in order to propose an architecture based on context-aware technologies.

1. Context-Aware Technologies

Context-aware applications could be defined as computing applications that use context information in order to automatically adapt their behavior to match the situation. Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between an user and an application. Context information can be gathered from a variety of sources (sensors, profiles, applications that report their current state and data interpretation services).

Context-aware technologies are able to support smart health-care environments and also user tasks. Some common applications related to that kind of situations are:

- **Support for social interactions**
  - assistance with everyday tasks;
  - dynamic organization of groups based on what activities they are interested in;
  - awareness of people activities in homes.
- **Smart spaces**

  Automatic configuration and reconfiguration of devices, in order to adapt their behavior to user needs and preferences.
- **Multi-modal technologies**

  They assist people by providing interaction with common devices (phones, televisions, etc.).
- **Health care applications**

  They provide health monitoring, accident monitoring, behavioral trend monitoring and cognitive health monitoring. Other health care applications provide reminders for treatments, taking medications, etc.

Services are able to modify their behaviour and the information provided, based on the emergency context in which the user is. Users can be notified in case of an emergency (fires, accidents, robberies, etc.) and be informed on how to proceed in those situations. When using mobile systems, the application's behaviour has to be adapted according to the changing situation. Some important aspects are network variations (bandwidth, latency, etc.), hardware variation (screen size, buttons, etc.) and software variations (memory capacity, installed applications, etc.).

Context Aware Computing relies on several independent enabling technologies. For input it relies on sensors, and these have their own limitations. For processing it relies on hardware, it depends heavily on artificial intelligence principles and those depend on data and rules extracted from knowledge of the world.

1.1 Common context-aware components

The components within the Context Manager Client and Context Manager Server (Context Broker) can be implemented using different technologies. As long as external interfaces and protocols comply with the REST-like and ContextML paradigm. In particular, these components, can be implemented using Java Enterprise technology. Technologies related to context-aware development, are an important way to implement functionalities regarded to context-aware applications.

Context-aware systems can be implemented in many ways. The approach depends on special requirements and conditions such as the location of sensors (local or remote), the amount of possible users, the available resources or possible extensions of the system. The way in which context-data is retrieved, predefines the architectural system design, based on available context-aware technologies.

1.1.1 Web 2.0

Web 2.0 was defined as “a trend economic, social and technologic that is building the base of new internet generation”, such trend is becoming more mature and characterized by user participation. *Web 2.0* considers the user as an information provider, and uses network as a platform to deliver or receive applications through a browser.

It has a complex and growing technology that includes server-software, protocols, standards-based browsers and various client-applications. A web 2.0 website may include following techniques: Rich Internet application techniques (AJAX), Cascading Style Sheet (CSS), XHTML markup and Microformats, organization of data in RSS, meaningful URLs, blog publishing and mashups, REST or XML Webservice APIs.

1.1.2 REST (Representational State Transfer)

REST is a programming style for hypermedia distributed systems. Each resource should be reached by an universal syntax used to define a specific hyperlink. Separation between client and server and the stateless interaction included in REST, make easier the component implementation and also reduce the complexity. All this improves the performance and scalability of the server.

REST-style architectures consist of clients and servers. Clients initiate requests to servers; servers process requests and return appropriate responses. Requests and responses are built around the transfer of "representations" of "resources". A resource can be essentially any coherent and meaningful concept that may be addressed.

A client in a rest state is able to interact with its user, but creates no load and consumes no per-client storage on the set of servers or on the network. The client begins sending requests when it is ready to transition to a new state. RESTful applications maximize the use of the pre-existing, well-defined interface and other

built-in capabilities provided by the chosen network protocol, and minimize the addition of new application-specific features on top of it.

1.1.3 Social Networking

Applications related to social networks exploit available knowledge about the situation of the user (context) adapting their functionalities in order to better serve the user's needs, in order to produce more focused and useful recommendations. Social context information, considers the relationships existing among a group of people.

Some activities involve both people actually being at some specific location and people not being there, but sharing a social relationship with others. Mobile applications are a very good source of social context data, coming in the form of address books, buddy lists and agendas; mobile devices such as smartphones are designed for fast and easy user interaction with these data. Location technologies available in many mobile platforms and devices (GPS, GSM cell-based location, Wi-Fi, Bluetooth, RFIDs) can detect the peers in the vicinity of a person and match/filter them on the social network data available to the mobile environment, in order to allow group recognition.

1.1.4 Context Ontology

An ontology suggests a complete conceptual schema related to one specific domain; it is often regarded to a hierarchic data structure that contains all relevant entities, the relationship between them, rules, and the specific links of the domain. Ontologies are applied commonly in both the artificial intelligence field and the knowledge representation. Programs can use it for: inductive reasoning, classification and problem solving techniques, in order to facilitate communication and information exchange between different systems.

The context ontology provides a way for describing basic concepts including users, locations, activities and computational entities (services, applications and devices). It also defines a set of attributes (user's status) and relationships hold between these entities. Context-aware services are mainly interested in how the context can be used to analyze historical information in order to predicting future trends.

1.1.5 Rule Engine

Rule Engine assesses individual rules and determines their applicability in a specific case or situation. The purpose of such component is related to manage business logic, it is an advanced interpreter of "rules". Each rule is composed of two parts, one condition and one action: when the condition (for example: "operation_cost > 5000 euros") is verified, the action (for example: give discount 5%) is executed. Rule engine input is composed of both a set of rules and also a set of data. The output is defined from the information received as input.

A rule engine contains, a knowledge-base (rules represented in a computational format), and an execution engine. The execution engine takes as its input one or more elements of data, applies a set of rules to that data, and generates some form of output. Execution engines can be based on production rules, logical or axiomatic rules and hybrid rules. The last ones are related to three execution engine environments that are used to support the delivery of higher-level rules represented by using hybrid models (BPEL Execution Engines, GLEE and SAGE).

1.1.6 Recommendation Techniques

Recommendation systems tries to forecast objects (film, songs, books, news, web pages, etc.) to which an user could be interested in, based on some profile information. Recommendation systems combine information provided from user, with data gathered implicitly (for example: objects acquired in past). Such information is compared to similar information that belongs to other users, with the purpose of generating a list of recommended objects.

It provides data analyzing to perform correlations from the user profile in order to determine interesting services for him taking implicit interests inferred from the analysis of the data available. A system is context-aware if it uses context to provide relevant information and/or services to the user. The key goal of context-aware systems is to provide relevant information and/or services based on current user context.

Recommender systems use content-based reasoning methods. They are systems capable of helping people to quickly and easily find their way through large amounts of information by determining what is of interest to a user.

Both context-aware systems and recommender systems are used to provide users with relevant information and/or services; the first based on the user's context; the second based on the user's interests. In some applications, location is used as a hard criteria to select relevant services that are close to the user; the predicted interest of the user is used as a soft criterion, just like some other contextual factors.

1.1.7 Data Mining

Data Mining consists of gathering useful information executed in automatic or semi-automatic way from big amounts of data. Techniques and algorithms of Data Mining have as main purpose analyzing many set of data in order to identify interesting regularities, known as patterns.

Due to dynamic nature of environment, data must be interpreted differently depending on situation (context). Implicit context-aware factors could be used to interpret and enhance explicit user inputs and thereby affecting data mining results to deliver accurate and precise prediction results. Data mining is a process that discovers patterns in data that may be used for valid predictions. Such process can filter useful and interesting context factors, produce accurate and precise prediction using those factors.

2. Technological Approach

In this section we introduce a suitable and reliable technological approach in order to implement the adaptation. In detail, we are going to describe the main functional components of this approach, represented by three logical levels.
Mobile devices can be used by network operators and service providers to learn more about user's environment, habits and preferences and to take advantage of this information for service personalization. Such information can be gathered through two means: direct feedback/input (application monitoring) or based on some analysis and post-processing. Analysis and post-processing information consider the usage of reasoning and learning tools for implicit information computation.

A technological approach to be used for supporting context-aware functionalities (gathering user context and implementing adaptation) is characterized by the next three logical levels (figure 1):

- Context Data;
- Context Analysis;
- Service Integration.

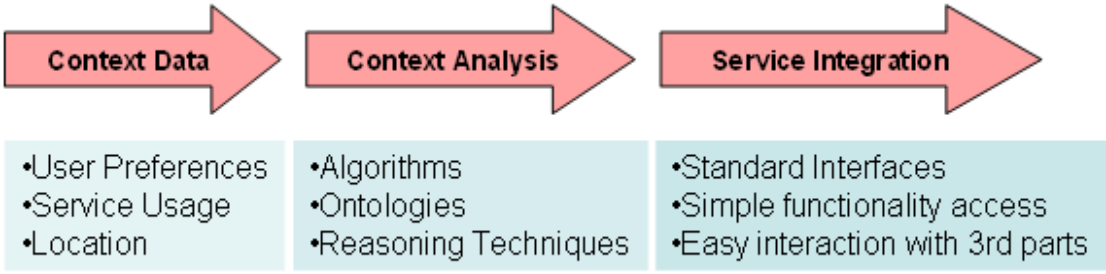

Figure 1: Logical Levels

2.1 Context Data Layer

It is the level in which the basic context information, generated from several informative sources available (context providers, data net providers, device info, user info, sensors, etc.) is retrieved in order to be further processed.
Inside of the Data Context level, it is possible to identify a heterogeneous variety of systems and platforms able to provide context information. Other systems are instead partially, or completely, outside of the operator domain. The systems outside of the operator domain belong to third parts that provide services or different kind of information (Google calendar, email Yahoo, etc.). The initial platform requirements of this level regard:

- Ability to collect context data in a fast and economic way. Since it is needed a high scalability, the Context Data Layer is a critical level from the performance point of view.

- Flexibility for integrating heterogeneous informative sources and consequently to support different kind of protocols and data formats.

2.2 Context Analysis Layer

The aim of this document is to describe and analyze advanced technologies to adapt the service on the user side, in function of the attributes that identify the context [described in task 2.1]. In the document D2.1 we analyzed the language for the description of services and context; and the description of possible adaptation policies. It is the level in which the context data, retrieved from the Context Providers, are processed. The main functions of this level are:

- Context identification of the service provided to the customer;
- Generation of more useful information based on the initial context information.

The initial platform requirements of this level are characterized by the ability to:

- perform *Reasoning* on the context information in order to obtain new information with a highest level of abstraction, regarding the initial information. *Context Analysis* is a level able to extract more information from initial context data, and is therefore necessary to support different processing techniques (statistics, semantic, etc.).
- apply recommendation techniques through a *recommendation engine* that uses multiple prediction strategies to predict how interesting a specific service is for the user.
  A recommender system compares the user's profile to specific characteristics, and looks for predicting the “rating” that a user would give to an item not yet considered. These characteristics may be taken from the information item or the user's social environment.
- support information representation that is shared between several members of the system and eventually with third parts.

2.2.1 Reasoning Process

The reasoning mechanisms are based on the next six steps [1], defined as (Figure 2):

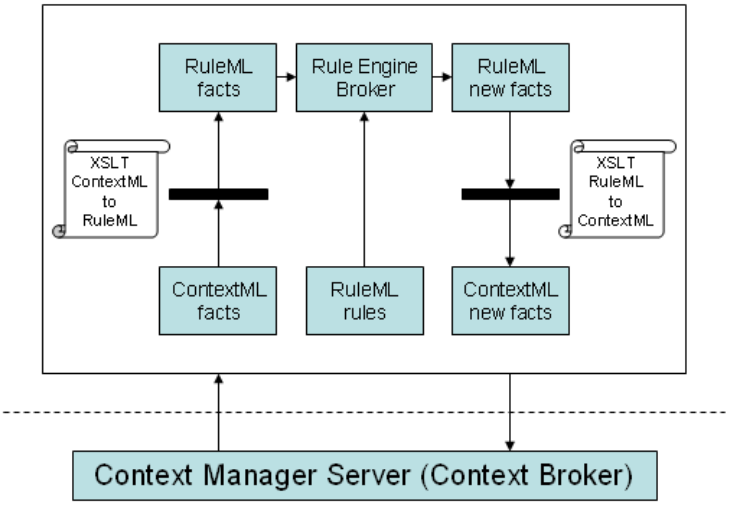

Figure 2. Situation Provider (reasoning process)

1. Specification of RuleML rules for situation inference;
2. Facts extraction from the ContextManagerServer (Context Broker);
3. Knowledge Base population through facts (translation ContextML to RuleML through an XSL stylesheet);
4. Execution of rules (knowledge base) in the Rule Engine Broker;
5. Possible assertion of new facts in RuleML, depending on executed rules;
6. Internal notification of the newly deduced facts to the Situation Provider.

The Situation Provider is a Context Provider often developed as a Stateless EJB and run on a JBoss Application Server. Situation information is computed by both interacting remotely with the ContextManagerServer (Context Broker) to retrieve context information from external Context Providers, and interacting locally with the Rule Engine Broker for situation inference.
The Rule Engine Broker integrates multiple rule engines, like JESS [4], and provides a single abstract interface based on RuleML. Most Java-based rules engines provide a technical call-level interface, based on a standard application programming interface (API), in order to allow integration with different applications. Many of them allow service-oriented integrations through Web-based standards such as WSDL and SOAP.
There is no standard language for writing rules. Many engines use a Java-like syntax, while some allow the definition of custom business languages. There are a number of different types of rule engines. Most Rules Engines are forward chaining, which can be divided according to:

- Inference rules: They represent behaviors of the type IF condition THEN action. For example, "Should this customer be allowed a mortgage?" → "IF some-condition THEN allow-customer-a-mortgage".
- Event Condition Action rules: The rule engines detect and react to incoming events. A rule engine of this kind could be used to alert a manager when certain items are out of stock.

A third class of Rules Engine might be called a deterministic engine. These Rules Engines may use domain-specific language approaches to better describe policies. This approach is often easier to implement and maintain, and often provides performance advantages over forward or backward chaining systems.

2.2.1.1 Retrieving context information

The Situation Provider component interacts with the ContextManagerServer (Context Broker) to retrieve useful context information in order to establish the main characteristics related to the current situation. The following context providers are involved:

- Context History: location-based information retrieved from GSM cell, GPS coordinates and civil location;
- Bluetooth devices identifiers provided by the local device;
- User profile;
- Calendar Provider is related to information about meeting's name, location and participants' email addresses;
- Address Book.

2.2.1.2 Providing situation information

Most context-aware approaches provide external interfaces by implementing REST-like interfaces through HTTP GET requests. A servlet will receive the HTTP GET request and return the ContextML content. When a remote HTTP invocation is executed, the servlet retrieves an instance of a Situation bean implementing the situation reasoning process, in order to infer the entity's situation at runtime. In case there are new facts, they are converted back into ContextML format from RuleML using a second XSL stylesheet to output the inferred situation.

Notifications are provided as external HTTP requests (for example to activate a service) or through a SIP PUBLISH towards a predefined SIP Presence Server acting as "Presence Network Agent" for integration into IMS-based networks.

2.2.2 Recommendation

The recommender system [5] compares the collected data to similar data collected from others and calculates a list of recommended items for the user. One of the most commonly used algorithms in recommender systems is Nearest Neighborhood approach. In a social network, a particular user's neighborhood with similar taste or interest can be found by calculating Pearson Correlation, by collecting the preference data of top-N nearest neighbors of the particular user (weighted by similarity), the user's preference can be predicted by calculating the data using certain techniques.
Another family of algorithms that is widely used in recommender systems is Collaborative Filtering. One of the most common types of Collaborative Filtering is item-to-item collaborative filtering (people who buy x also buy y), an algorithm popularized by Amazon.com's recommender system. Some authors think that efforts in this field, should be concentrated in deriving substantially different approaches, rather than refining a single technique. A good solution is a combination of many methods.
A prediction strategy combines multiple prediction techniques by deciding which prediction techniques are the most suitable to provide a prediction based on the most up-to-date knowledge about the current user, other users, other information

items and the system itself. Current prediction methods include social filtering, case-based reasoning, item-item filtering [2] and category learning [3]. For different kind of data contexts, different prediction strategies can be defined in the engine. In some approaches, prediction techniques have been developed according to contextual factors.

2.3 Service Integration Layer

In this level the Context-Aware functionalities are exposed towards both the service platforms and the third applicative parts. The initial platform requirements at this level are:

- Ability to offer standard interfaces adapted to different kind of application environments;
- APIs capable to provide simple access to every functionality related to the other levels;
- Easy interaction with third parts in order to enable new business models.

2.4 External Systems

They are placed inside of the two logical levels of Context Awareness platform: Context Data and Service Integration. Inside of the Service Integration level, it is possible to identify an suitable environment in order to create and execute services. In this level are typically present software platforms defined as Application Servers. The main modalities that are able to connect these systems are based on Web (WebServices, REST) or Java technologies (J2EE).

In many context-aware architectures, user information is exchanged between client and server-side following a REST-like paradigm based on HTTP and XML, which facilitates the exchange of information with mobile devices. Requests are typically HTTP GET or POST requests that return XML-based content (the user information) according to the ContextML language.

Representational state transfer (REST) is a style of software architecture for distributed hypermedia systems such as the World Wide Web. The systems that implement the REST principles are defined as " RESTful". REST-style architectures consist of clients and servers. Clients initiate requests to servers; servers process requests and return appropriate responses. Requests and responses are built around the transfer of "representations" of "resources". A resource can be essentially any coherent and meaningful concept that may be addressed. A representation of a resource is typically a document that captures the current or intended state of a resource.

The main goals of REST include: Scalability of component interactions; generality of interfaces; independent deployment of components; intermediary components to reduce latency, enforce security and encapsulate legacy systems. REST's client-server separation of concerns simplifies component implementation, reduces the complexity of connector semantics, improves the effectiveness of performance tuning, and increases the scalability of pure server components. Layered system

constraints, allow intermediaries to be introduced at various points in the communication without changing the interfaces between components, thus allowing them to assist in communication translation or improve performance.

REST enables intermediate processing by constraining messages to be self-descriptive: interaction is stateless between requests, and standard methods and media types are used to indicate semantics and exchange information.

Appendix A Acronyms Table

XML	Extensible Markup Language
HTML	HyperText Markup Language
XHTML	Extensible HyperText Markup Language
JSON	JavaScript Object Notation
DHTML	Dynamic HyperText Markup Language
CSS	Cascading Style Sheets
J2ME	Java 2 Platform Micro Edition
GUI	Graphic User Interface

Figures Index

Bibliography

[1] Goix, L., Valla, M., Cerami L., Falcarin P.; Situation Inference for Mobile Users: a Rule Based Approach. *Telecom Italia Lab, Politecnico di Torino, Italy*

[2] Herlocker, J., Konstan, J.; 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