PICO

DELIVERABLE 2.1

Context and Service Adaptation Policies

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 language for the description of context data and service adaptation policies.

CHANGE LOG

Version	Date	Description
1.0	18/11/2009	This is the first draft of the Deliverable 2.1
2.0		Final version

TABLE OF CONTENTS

Contents

Introduction

The aim of this document is to describe the context definition and the context reasoning process applied to adapt the service. This description consists in identifying the most recurring attributes capable of characterizing a user context based on terminal (processing power, memory, user interface), network (available bandwidth, quality of service, protocol) and location (language, coordinates, etc). In this document we will also introduce the most important components related to the context reasoning process, used to continually adapt the service.

1. User Context Definition

Human-computer interaction is an important concept that does necessary the introduction of context aware computing in which the idea is to adapt the computer and its behaviour to the user context. There are a lot of definitions of user context, in general, user context is defined as “any information that can be used to characterize the situation of an entity (person, place, object, etc)”.

The user context is especially dynamic for mobile systems (mobile phones, laptop, etc). When using such devices, it is important to adapt the application’s behaviour according to the changing situation.

- **Environmental Factors**

There are different approaches to choose the relevant aspects in environmental context. For example, network variations (bandwidth, latency, etc.), hardware variation (screen size, buttons, etc.), memory and software variations (memory capacity, installed applications, etc.).

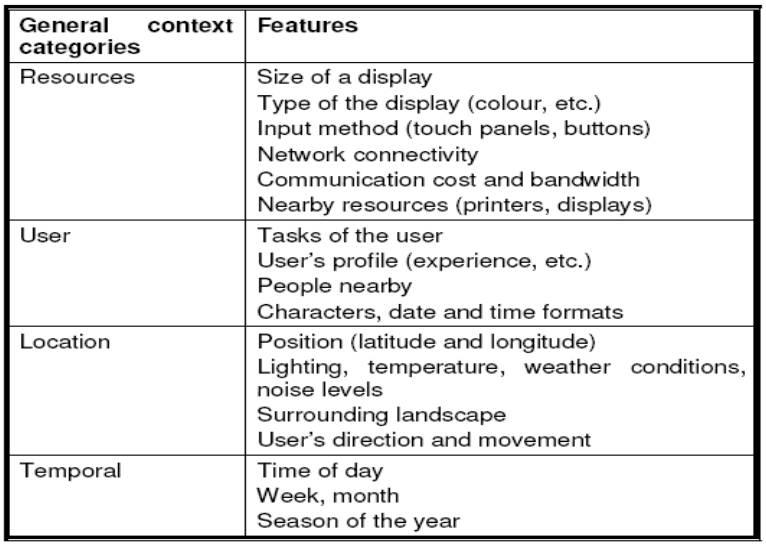
Information about the infrastructure and the location (physical and logical, at home, at work, etc.) is also very relevant for the user context. The following kinds of environmental context can be considered:

- o **Resource Context:** Refers to the demand of multi-delivery and thus includes information about the relevant devices, device classes, documents, network, available services, etc.

- o **User Context:** Takes into account personal information about the user and user’s classes as user's identity, characteristics, capabilities, universal preferences, the state of the user, information about his or her main activity, etc.

- o **Location Context:** Describes the geographical coordinates, identity and the state of the location (the people present at that location, etc). The location context includes also aspects of the perception of the physical characteristics of the location (e.g. temperature, brightness, noise levels, etc).

- o **Temporal Context:** The temporal context (e.g. the absolute time, hour, am/pm, etc.) allows to adapt the application with regard to certain timing constraints such as the hour to go to work.



General context categories	Features
Resources	Size of a display Type of the display (colour, etc.) Input method (touch panels, buttons) Network connectivity Communication cost and bandwidth Nearby resources (printers, displays)
User	Tasks of the user User's profile (experience, etc.) People nearby Characters, date and time formats
Location	Position (latitude and longitude) Lighting, temperature, weather conditions, noise levels Surrounding landscape User's direction and movement
Temporal	Time of day Week, month Season of the year

Table 1. Categorization of environmental context

User context information considers also:

1. Profile: Defines the basic user information related to static and persistent data (e.g. gender, name, address, etc). According to the different characteristics of the profile, different methods may be integrated to model the user. There are several possible approaches to represent the user profile (user model). Rule based is the most common method to represent the user model ("If ... Then take action").

2. Preferences: It is related to all the raw information about what the user likes/dislikes, based on direct feedback or input, or on postprocessing/learning.

3. Service Usage History: How the service is used and the history of usage. The system can learn about the user behaviour in order to provide more service personalization.

4. Service Profile: Which service is used by which user, potentially in which context and with which parameters/values (e.g., service name, version, active status).

5. Device usage: What the user is doing on his device. It is related to applications executed on pc/mobile phone, etc.

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.

In order to extract a higher-level description of the user context, data only available on the device can be aggregated with information only available to mobile operators and service providers.

1.1 Data Format and Access

Definition and representation is fundamental to communicate and process context information. Communicating simple context over wireless networks between different kind of devices requires lightweight representation. On the other hand, reasoning about situations and environments requires robust representation and models.

Context-aware applications need a way to recognize and represent context in order to apply reasoning and adaptation policies. Context is a form of metadata that is used to describe situations and is most often associated with a temporal event or state. The processing and manipulation of this metadata is fundamental to achieve interoperability between providers from different domains.

In [1] authors observe that context representation for demanding reasoning requires that the modelling approach should be able to: allow partial validation independently of complex interrelationships; enable rich expressiveness and formalism for shared understanding; indicate richness and quality of information.

Context Meta Language (ContextML) is an XML-based language that enables context-aware platforms to discover the user information they need. More heavyweight database models and reasoning engines use it, in order to send and receive context. ContextML is used to encode not only the context data but control messages as well.

1.1.1 Context Representation

The ContextML schema is composed by:

- **ctxEls**: Contains a set of information for a certain entity.
- **ctxAdvs**: Contains the advertisement of provider features to the broker.
- **scopeEls**: Contains response from the Broker to the getAvailablesAtomicScopes method.
- **ctxPrvEls**: Contains response from the Broker to the getContextProviders method.

Any user information given by a provider is characterized by an entity and a specific scope. When a provider is queried, it returns the required data in a XML document, which contains the following elements:

- **contextProvider**: A unique identifier for the provider of the data.
- **entity**: The identifier of the entity which the data are related to.
- **scope**: The scope which the context data belongs to.
- **timestamp and expires**: respectively, the time in which the response was created, and the expiration time of the data part.

- **dataPart**: Part of the document which contains actual user information data which are represented by a list of features and relative values through the *<par>* element ("parameter"). They can be grouped through the *<parS>* ("parameter struct") or *<parA>* ("parameter array") elements if necessary.

For example, we can define a *getCivilAddress* method of the Location Provider for latitude 45.112 and Longitude 7.67 for user "jack" that is invoked through:

```
GET
http://myServer/getCivilAddress?username=jack&lat=45.112&lon=7.67
HTTP/1.0
```

And returns the following XML content:

```
<?xml version="1.0" encoding="UTF-8" ?>
<contextML xmlns="http://ContextML/1.1"
xmlns:xsi="http://www.w3.org/2001/XMLSchemainstance"
xsi:schemaLocation="http://cark3.cselt.it/schemas/ContextML-
1.1.xsd">
    <ctxEls>
        <ctxEl>
         <contextProvider id="LP" v="1.0.2" />
         <entity id="jack" type="username" />
         <scope>civilAddress</scope>
         <timestamp>2009-01-24T16:05:19+01:00</timestamp>
         <expires>2009-01-24T17:05:19+01:00</expires>
         <dataPart>
             <parS n="civilAddress">
              <par n="street">Main Street</par>
              <par n="postalCode">10148</par>
              <par n="city">Torino</par>
              <par n="subdivision">TO</par>
              <par n="country">Italy</par>
             </parS>
         </dataPart>
        </ctxEl>
    </ctxEls>
  </contextML>
```

1.1.2 Scopes

User information is subdivided into scopes, namely sets related to the same information category. For example, the scope named "position" groups latitude, longitude and range regarding a certain entity's location. Scopes can be atomic or aggregated, as union of different atomic scopes.

Since ContextML is a meta-language, it does not require any schema (XSD), XML tags or structure for each specific information domain or concept, but is rather based on the idea of "scopes", which are abstract definition of a coherent set of information. To each specific domain of information is associated one or more scopes. A scope is a simple table of concepts, grouped together and identified by a name, which is used as parameter name (n attribute of <par> element) in ContextML.

The scopes currently defined are location and device information. There are defined 3 scopes related to location information, "*cell*" for GSM-based cell information, "*position*" for GPS-related information, and "*civilAddress*" for civic information. Some examples are:

Scope name: Position

Description: Information related to the geographical position of an entity

Parameter name	XML schema data type	Description	Required
latitude	xs:float	WGS84 lat	YES
longitude	xs:float	WGS84 lon	YES
accuracy	xs:int	Range in mts	YES
locMode	xs:string	positioning technique used ("IP", "GPS", etc)	YES
altitude	xs:float	Altitude in mts	NO

Scope name: DeviceInfo

Description: Static information related to the device

Parameter name	XML schema data type	Description	Required
memory	xs:double	Amount of total RAM (in bytes)	NO
imei	xs:double	Phone IMEI	NO
osType	xs:string	Operating System short name	NO
cpu	xs:string	Number of CPUs	NO
screenH	xs:integer	Screen height (in pixels)	NO
screenW	xs:integer	Screen width (in	NO

		pixels)	
screenColor Bits	xs:integer	Color Deep	NO

2. Service Adaptation

The main goal of a context-aware application or service is being able to change its behaviour according to a context change. Context-aware systems are aware of their environment in order to react intelligently. Adaptation is a very important element of a context-aware system. There is an increasing interest in context-aware systems that interact with external context factors (location, presence, temperature, etc). Every service should be able to access the context information to adapt itself according to the user situation, or to trigger some actions whenever some states are detected.

Service adaptation is considered the set of mechanisms that allow the personalization of the services. In a context-aware platform, these mechanisms are important in the execution of the services. Static (user profile) and dynamic (context) information, are the way to provide users with services based on their needs, preferences, interests or expertise.

Common context-aware architectures propose to save multiple versions of the service and then send to the user a version according to his context, in order to achieve the service adaptation to the context changes. These architectures aims to:

- realize dynamic services adaptation according to the user's context. Each service is presented by a set of components that are added, removed or replaced according to profiles incompatibility.
- dynamically adapt services to the user's context where each service is composed of a set of interconnected components. This adaptation aims to select the service components depending on the context.
  The adaptation of the service at the time of its execution consists of dynamically changing the service composition following the changes of the environment by adding, removing or replacing service components.
- provide adaptable services to mobile users. This kind of architecture addresses simultaneously the logic service adaptation and the multimedia content adaptation.

Researches distinguish between two types of adaptation:

- Logic adaptation: Several researchers focus on the logic adaptation of services [3][2]; in this case the service is represented by a set of components; the adaptation aims at changing these components by adding or replacing a component.
- Content adaptation: Other researchers [5] concentrate on the content adaptation; it is a set of adaptability forms that chooses changes of the content provided by the service depending on the context. A typical form of the content adaptation is changing the encoding of a multimedia stream or changing the presentation of a service depending on the context.
  Content adaptation aims to change the properties of the data presented to the user. The service content adaptation, adapts the data service to the terminal capabilities, to the network capabilities and/or to the user preferences.

According to the requirements and characteristics of PICO project, service adaptation will be based on content. A service is a combination of data and operations, so the adaptation of the content of a service is the adaptation of services data according to the user's context.

Adaptation [1] can take place according to the next different levels: Technology, service behavior, user interface, presentation and content. The five different policy levels of service adaptation are:

- **Technology**

  In this level, the service is adapted to the technology context. For example the adaptation policies can be based on:

  - information is encoded for specific mobile devices with different characteristics (display size and resolution, memory, CPU power, etc.); or
  - specific network conditions (reducing image resolution and color depth, and lowering video frame rate to match the network bandwidth).

- **Service behavior**

  It may be adapted according to the user's location or task. An example of this kind of adaptation based on the user's location is:

  - the zone alert feature of a tracking service, where the alert message is sent to specific people when the tracking target leaves or enters a pre-defined zone.

  On the other hand, it may also be adapted according to the user's tasks. For example:

  - a driver may define that he wants text information to be translated to voice while driving.

- **User interface**

  It may be adapted according to the user's tasks, the system in use (device and network), and the user's physical conditions. For example:

  - the user interface is changed from graphic user interface to voice interface when a blind user is accessing the service or when a user is driving.

  The service adaptation in the user interface level requires that the terminal can support different kinds of user interfaces; otherwise adaptation in the user interface level is limited or not possible.

- **Presentation**

  The service visualization may be adapted to the user's tasks, social aspect, and physical condition. For example:
  - the visualization of the service may be adapted according to the user's age [4].

- **Content**

  In this level the content of the service is adapted to the current location, situation and user. For example:

  - age groups.
  - gender.
  - preferences.

2.1 Context Reasoning Process

The goal of context reasoning is to deduce high-level, implicit context from low-level, explicit context. The structure of the context predicate has tree fields: a subject that is a contextual entity, an object that is contextual entity or a datatype value, and a verb that describes the attributes of the subject or the relationship between the subject and the object.

The reasoning process consists in the inference of a high-level user's situation by the means of rule-based reasoning techniques applied to user profile and context data. Inferred information can be used to adapt existing services and to offer new and more powerful ones.

2.1.1 Identifying Situations

Context Reasoning can be defined as the process used to derive logical location and social state from context data, and situation reasoning as the identification of the user's activity from context data, logical location and social state. In [6] authors define "situation" as the grouping of three high-level concepts that can easily be used for service personalization:

2.1.1.1 Logical Location

It associates a meaning to the place where the user is currently located. Such location can be absolute or relative, for example when moving in a vehicle. In case of absolute position, GPS coordinates and civil location are associated to the type of place (office, meeting_room, home). User's location is an important criterion in service adaptation for context-aware systems, location contexts are very useful in location-based emergency services, for example, to find out the closest ambulance in order to provide fast medical assistance.

2.1.1.2 Activity

It is related to the user's current activity (working, cooking, waiting_formal_meeting). High-level contexts such as "what the user is doing" are often derived from relevant environmental contexts. By identifying the current user activity, the system can adapt the service according to the context retrieved. Reasoning mechanisms, focused on this kind of situation, are characterized by applying inference over relationships between entities.

2.1.1.3 Social state

It provides a way to recognize in which kind of social environment the user is. It is possible by providing eventual relationships between entities (user_with_colleagues, user_with_friends, user_with_jack, etc).

In case a more specific representation of the situation is needed, it is important defining more specific rules that will be necessary for reasoning on each concept of interest. According to authors in [6], it is possible characterizing a list of target situations for a user, for example, when the user has a meeting scheduled in his Calendar, a set of rules can automatically deduce his current activity (*waiting_formal_meeting*, *late_to_meeting* or *formal_meeting*).

User's status often depends on various kind of contextual knowledge, for example, a GPS sensor provides user-location, a calendar provides the information about scheduled activities, time, etc. A reasoning mechanism is able to establish different kind of relationships between context activities and events. Context data can be combined in order to further deduce more complex high-level context. Let's consider the *waiting_formal_meeting* activity, it will be inferred only if:

- The user is working,
- The user has scheduled a "meeting" event in his/her Calendar,
- The user is currently in a "meeting room" (logical location)
- The scheduled meeting place corresponds to the current user location,
- Some participants are still missing.

In case the user is not at the meeting place, his/her activity will be *late_to_meeting*. Then another service can be used to alert the user by SMS to remind him/her of the meeting and the expected place. Finally, if all meeting participants are present (Bluetooth devices inference), the user activity will be updated to *formal_meeting*.

2.1.2 Selecting the rule language

In order to both share the information (facts) within the reasoning module and infer situations, the ContextML data has to be express in a standard rule language. A rule engine, like JESS [7], is a software system that executes one or more rules in a runtime production environment. The rules might come from company policy ("All customers that spend more than $100 at one time will receive a 10% discount"), or other sources. A rule may detect that a specific situation has occurred and raise an event or create higher level knowledge.

A rule engine provides the ability to: register, define, classify, and manage all the rules, verify consistency of rules definitions ("Gold-level customers are eligible for free shipping when order quantity > 10" and "maximum order quantity for Gold-level customers = 10"), define the relationships between different rules, and relate some of these rules to IT applications that are affected or need to enforce one or more of the rules.

2.1.3 Asserting facts from context data

In order to apply rules on context, it has to be translated to facts. User's data represented by attribute-value couples can be translated in facts; for example the operative system version of user's phone can be written as a fact:

```
(assert (osVersion user 1.0))
```

In Jess language the first field acts as a category for the fact. In order to represent more complex and structured data as facts, there exists a technique known as "reification". By using this technique, complex data are represented with a set of *n* simple facts sharing the same symbol, which act as a category identifier for this set. For example: "For the user 1234 the connection with the cell 3400000000 represents a location type 'office' ". This information is represented by:

```
(assert (user location 1234))
(assert (user location 3400000000))
(assert (type location office))
```

Every time reification is necessary, a specific fact "identity" is added, in order to tie the category identifier (location) with the symbol that was referred to the user (1234).

```
(assert (identity 1234 location))
```

2.1.4 Designing Rules

Designing rules is a process that considers rules as: set of pre-conditions that include a logical location (meeting_room) and a social state (with_colleagues), in order to infer from them an Activity (informal_meeting). According to authors in [6], situation rules can be designed by:

```
(defrule [context pre-conditions] =>
     (assert ([new situation]))
     (retract ([no longer valid situation]))
```

Appendix A Acronyms Table

XML	Extensible Markup Language
HTML	HyperText Markup Language
JSON	JavaScript Object Notation
LAMP	Linux Apache MySQL Perl Python PHP
CSS	Cascading Style Sheets
J2ME	Java 2 Platform Micro Edition

Bibliography

[1] Context-based Service Adaptation Platform: Improving the User Experience towards Mobile Location Services, Saowanee Schou, *Center for Information and Communication Technologies, Technical University of Denmar*k.

[2] B. Marquet, C. Gustave, A. Lefebvre, S. Nemchenko, S. Chassande-Barrioz, “Secured services in a multi-tier architecture”, In World Telecommunications Congress, WTC 2002, Paris, September 2002.

[3] D M. BRAIN, “Concepts for Service adaptation, scalability and QoS handling on mobility enabled networks”, IST-1999-10050 project, deliverable 1.2., 31 Mars 2001. [9] J. G. Kim, S. F. Wang Y and Chang, “Content-Adaptive Utility Based Video Adaptation”, In ICME2003.

[4] Nivala, A-M. and Sarjakoski, L. T.(2004) Preventing Interruptions in Mobile Map Reading Process by Personalization. In: Proceedings of The 3rd Workshop on 'HCI in Mobile Guides', in adjunction to: MobileHCI04, 6th International Conference on Human Computer Interaction with Mobile Devices and Services, September 13-16, 2004, Glasgow, Scotland.

[5] L. Boszormenyi, H. Hellwagner, H. Kosch, M. Libsie, S. Podlipnig, “Metadata driven adaptation in the ADMITS project”, In EURASIP Signal Processing: Image Communication Journal, Vol. 18, No. 8, September 2003, pp. 749-766.

[6] 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*

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

[8] RuleML: The Rule Markup Initiative, http://www.ruleml.org/