Contents

Abstract

The objective of this internship is to develop the approach of structural adaptation of a software component to manage resources limited effectively in the ubiquitous environments. So we first present the theory of Ubiquitous computing, context-Aware and a context-aware framework: Toolkit in the report. Then we present a scenario of ubiquitous application in the building site.

Thanks

I wish to thanks M. Abdelhak-Djamel SERIAI, teacher-researcher in ENSM-Douai, who accommodated me in the laboratory csl and directed my research task. I make a point here of testifying to him all my recognitions for his assistance, his availability, and his encouragement throughout this internship.

I wish to thanks Gautier BASTIDE, M.SERIAI's assistant, for his assistance, his remarks and the discussions which we had together.

I wish to thank Ecole des Mines de douai who accommodates me within the department of integrated manufacturing and computer science in a very pleasant and convivial environment.

At last, I wish to thank all the people who contributed to the development of this project.

Introduction

Comparing with computing modes before, In the Ubiquitous computing age, the computer has largely changed his role and action. In the environment of ubiquitous computing, people can use several computing equipments anytime and anywhere. So the user is able to operate all the equipments. Therefore ubiquitous computing emphasizes the user' intent and satisfaction; it's a human-centered computing paradigm.

In the environment of ubiquitous computing, the computer is weaved into the everyday life to enable users to get computing and information services anytime and anywhere. When the users use the computer, their attention is often focused on the tasks rather than computing system. The computer has to know how and when to offer the information to the user automatically. In this situation, the context is the effective and even the only basis for the computers to determine their behaviour. So context-aware is one of the important technologies of ubiquitous computing.

Chapter1. Theory: Ubiquitous computing and Context-Aware

1. Ubiquitous computing

Ubiquitous computing is a human-centered computing paradigm, which emphasizes that computation should cater for user's intent and coordinate with users automatically to let them concentrate on their tasks. In the Ubiquitous computing environment, computation should be weaved into everyday life to enable users to get computing and information services anytime and anywhere.

1.1. Defining ubiquitous computing

The vision of ubiquitous computing-first expressed by Weiser and grounded in experimental work done at Xerox PARC-holds the promise of yet another interaction paradigm shift. Thereafter, Mark Weiser expounded ubiquitous computing more systematic and comprehensive in his paper "The Computer for the 21st Century" in 1991: The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it [11]. He insisted that computation should cater for user's intent and coordinate with users automatically to let them concentrate on their tasks. In the Ubiquitous computing environment, computation should be weaved into the everyday life to enable users to get computing and information services anytime and anywhere. Ubiquitous computing integrates computation into the environment, rather than having computers which are distinct objects. Promoters of this idea hope that embedding computation into the environment would enable people to move around and interact with computers more naturally than they currently do.

1.2. Characteristic of ubiquitous computing

How to satisfy the user's intent in the environment of ubiquitous computing? In the paper "The Computer for the 21st Century", Mark Weiser conceived a lady (Sal)'s life scene in the environment of ubiquitous computing. We can understand the characteristics of ubiquitous computing through these scenes.

At first, the computing can know the user's intent automatically or through the interaction with user: when Sal wakes up, the computing has prepared a cup of coffee for her according to her habit; when Sal looks out of her windows about the environment, she can see the environment; when Sal Glance at the windows to her kids' rooms, she can know the kid's activities.

Ubiquitous computing has to understand the user's tasks and find out the best solution in the information according to the user's direction: when Sal wakes up, the computing has prepared a cup of coffee for her according to her habit.

After finding the solution, the problem is how to present the solution: There are many methods to present the information. In Sal's life, most of the information is offered to her by the vision equipments, for example, "windows", "foreview mirror", "telltale" and "tab". Meanwhile there are many interaction and cooperation among these computing equipments in the environment of ubiquitous computing, and the challenge is how these computing equipments offer the information to users harmonically: there are several "windows" in Sal's room, the information Sal needs can be presented no conflictive by these vision equipments.

We can conclude the characteristics of ubiquitous computing as follows:

- Ubiquitous: access anything by anybody via any devices anywhere and anytime.
- Invisibility: invisibility computing, computation should be weaved into the everyday life naturally, the computer disappear from user's sight.
- Dynamic: usually the user's movement in the environment of ubiquitous computing results in the change of users' set in certain space, and the mobile equipments can enter or exit computing dynamically which results in the change of computing structure.
- Adaptation: the computing can apperceive or conclude the user's intent and offer the information to the user automatically.

2. Context-aware

The human computer interaction can be improved with context information. With help of such information computer systems and applications can be made more users friendly and flexible. Especially in the mobile environment where the environment and user's needs change rapidly the use of context information will be important.

2.1. Defining context

For definition, context is the "whole situation, background or Environment relevant to some happening or personality." This definition is too general to be useful in context-aware computing.

In the work that first defined the term "context-aware", Schilit and Theimer refer to context as location, identities of nearby people and objects and changes to those objects (Schilit and Theimer, 1994) [9]. In a similar definition, Brown, Bovey, & Chen define context as location, identities of the people around the user, the time of day, season, temperature, etc. (Brown et al.,1997)[2]. Ryan, Pascoe, & Morse define context as the user's location, environment, identity and time (Ryan et al., 1997) [7]. Dey enumerates context as the user's emotional state, focus of attention, location and orientation, date and time, objects and people in the user's environment (Dey, 1998)[3].These definitions that define context by example are difficult to apply. When considering a potential new type of context information, it is not clear how the definition can help us decide whether to classify the information as context or not. For example, none of the previous definitions helps decide whether a user's preferences or interests are context information.

Other definitions have provided synonyms for context, referring to context as the environment or situation. Some consider context to be the user's environment, while others consider it to be the application's environment. For example, Brown defines context to be the elements of the user's environment that the computer knows about (Brown, 1996) [1]. Franklin & Flaschbart see it as the situation of the user (Franklin and Flaschbart, 1998) [5]. Ward, Jones, & Hopper view context as the state of the application's surroundings (Ward et al., 1997) [10], and Rodden et al. define it to be the application's setting (Rodden et al., 1998) [8]. Hull, Neaves, & Bedford-Roberts include the entire environment by defining context to be aspects of the current situation (Hull et al., 1997) [6]. These definitions are clearly more general than enumerations, but this generality is also a limitation. These latter definitions provide little guidance to analyze the constituent elements of context, much less identify them.

Based on these prior attempts to define context, we will proceed with the following definition, provided by Dey & Abowd (Dey and Abowd, 1999) [4]:

Context: any information that can be used to characterize the situation of an entity, where an entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves. Context is typically the location, identity and state of people, groups and computational and physical objects.

2.2. Classification of Context

We introduce four essential categories of context information —identity, location, status and time.

- Identity: refers to the ability to assign a unique identifier to an entity. The identifier has to be unique in the namespace that is used by the applications.
- Location: is more than just position information in 2-D space. It is expanded to include orientation and elevation, as well as all information that can be used to deduce spatial relationships between entities, such as co-location, proximity, or containment.
- Status: identifies intrinsic characteristics of the entity that can be sensed. By example, for a person, it can refer to physiological factors such as vital signs or tiredness, or the activity the person is involved in, such as reading or talking.
- Time: is context information as it helps characterize a situation. It is most often used in conjunction with other pieces of context, either as a timestamp or as a time span,

indicating an instant or period during which some other contextual information is known or relevant.

2.3. Representation of context

In the environment of ubiquitous computing, there are many types of equipment which can only get the local context information from the computing environment. It's necessary to communicate the information among them for representing user's context. Therefore, the most important thing is how to represent these different data with the same model and transfer them among the different equipments.

In order to define a data model for a context a data format capable of representing the information outlined in the previous section is required. There are a number of possible representations for this information. They range from primitive data types to full types and objects. Primitive types are flexible but, depending on the system requirements, they may not be efficient or concise enough for representing the desired information. A "simple" format is easier to make universal and there isn't a significant performance penalty for such an approach. The cost of a simple data representation, such as text, is mostly the expense of encoding and decoding.

2.4. Context-aware

Context-awareness means that one is able to use context information. A system is context-aware if it can extract, interpret and use context information and adapt its functionality to the current context of use. The challenge for such systems lies in the complexity of capturing, representing and processing contextual data. To capture context information generally some additional sensors and/or programs are required. To transfer the context information to applications and for different applications to be able to use the same context information a common representation format for such information should exist. In addition to being able to obtain the context-information, applications must include some "intelligence" to process the information and to deduce the meaning. This is probably the most challenging issue, since context is often indirect or deducible by combining different pieces of context information. The actions carried out by context-aware can be viewed as a three step process.

- Collect the information that relates to the context

  When the computing-context changes, a system must find out what new resources are available and what their characteristics and capabilities are by the use of simple sensors or resource manager.

- Decide which resources to use

  A system should be able to analyse the data collected and select the adaptation data based on the surrounding context. For example, where the user is located, and what co-located objects are present.

- Action

  At last, a system should be able to employ the resource chosen in the preceding step. The action should be able to modify the parameter, adapt the application, present the data to user and reform the context.

3. Toolkit

The Context Toolkit, a context-aware framework, takes a step towards a peer-to-peer architecture but it still needs a centralized discoverer where distributed sensor units (called widgets), interpreters and aggregators are registered in order to be found by client applications. The toolkit's object-oriented API provides a superclass called BaseObject which provides generic communications abilities to ease the creation of own components.

Toolkit uses an object-oriented approach in designing the architecture. The architecture consists of three main types of objects:

- Widget, implements the widget abstraction
- Server, responsible for aggregation of context
- Interpreter, responsible for interpretation of context

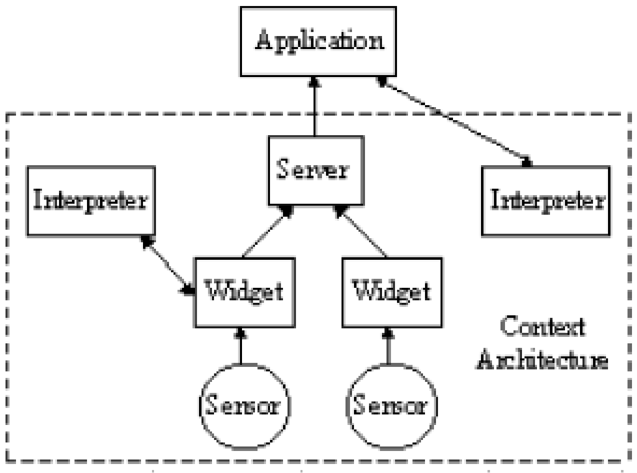


FIG.1- Architecture of Toolkit

This figure shows the relationship between the objects and an application. Each of these objects is autonomous in execution. They are instantiated independently of each other and execute in their own threads, supporting its requirement for independence.

3.1. Context widget

Context widgets are responsible for separating the details of sensing context from actually using it. They abstract away the details of how the context is sensed from applications and other context components that need the context. Each context widget is responsible for some small set of context that is captured from a (hardware or software) sensor.

A context widget is quite similar to a GUI widget. Just like a GUI widget hides the details of how the input is being captured (mouse, keyboard, pen, etc.) and simply provides information about interaction (button press, button release, etc.), a context widget hides the details of how a particular type of context is being captured or sensed from a software or hardware sensors and simply provides information about the context the sensor provides. For example, a web page that delivers current temperature or a thermometer outside your house can be used to provide temperature information. For the most part, an application does not care about the source of the temperature information or how it is being sensed. The context widget abstraction allows an application developer to simply use the temperature information without worrying about how it's being.

3.2. Context server

Context servers are responsible for aggregating all contexts about a particular entity: person, place, or object. We believe that entities are keys to the notion of context. A server supports this aggregation behavior by taking on all of the behaviors (attributes, callbacks, and services) of relevant context widgets. It subscribes to all the widgets that you think are relevant (when resource discovery is added, the relevant widgets will be determined automatically). For example, a server representing a particular person should subscribe to any widget that can provide information about that person. A server essentially behaves as a mediator, with whom an application can talk to and get all the benefits of the widgets that it encompasses.

3.2.1. Features

Servers are subclass of Widgets, so they have all the features of Widgets. As well, they act as an aggregator of context about an entity and act as a mediator between applications and widgets.

3.2.2. Aggregator

A server subscribes to all the widgets that are relevant to the entity the server represents. If a storage mechanism has been set up for the server to use (one exists, by default), the server acts as a central repository for all context known about its entity.

3.2.3. Mediator

A server subscribes to all the widgets that are relevant to the entity the server represents. When an application wants to poll for context, subscribe to context, or execute a particular service, it no longer needs to individually communicate with a widget, and instead, can talk to the relevant server.

The server acts as a mediator to all the widgets that can provide relevant context information about the server's entity.

3.3. Context interpreter

Context interpreters are responsible for interpreting or converting context from one form to another. They maintain no state (although they could) but simply take context in and output new context information. This can be as simple as taking in a name and returning the corresponding email address. It could be more complex and take in the number of people in a room, the relative gaze directions, the audio level, and the time of day, and return whether or not a meeting was occurring.

4. Conclusion

The majority work of application “context-aware” is placed in the environment of ubiquitous computing. First, we have defined the conception of ubiquitous computing and characteristic of ubiquitous computing. Context-aware computing is still in its infancy. We have attempted to clarify the notion of context, and laid out foundations for the design and development of context-aware applications. Then we have defined broad categories of entities capable of providing context information, and identified basic categories of context.

Chapter2. Scenario of Ubiquitous application

1. Introduction

It was easy to determine the execution context of an application before: the user is in the office in front of the computer. Now, with the development of technology, it's possible to carry out an application anywhere and anytime. The final mobiles are multiplied (PDA, smartphone). The user should be able to reach information where he is and when he wants. Because of new constraints related to the environment exist for example the limitation of the resources. In fact, all the terminals do not have the same characteristic. It's impossible to specify an application for a terminal. It is with the application to be adapted according to the conditions for implementation. It is thus essential to take into account information on the context of an application. For example, if one takes the case of an application of guide embarked in a car. When the user drives, the application must primarily give him vocal information whereas if it is with the stop of visual information can be communicated to him. The applications must be able to be adapted easily or to even adapt themselves to the evolutions of their context of execution with an aim of continuing function correctly and guaranting its performances and its quality of service. It must take account of new possibilities which could appear during its execution and which would involve an increase in its performances. It must be able to face problems that it would meet such as the reduction in the band-width of the network either the disconnection of a machine or a peripheral.

2. Application

We present an application scenario of building site which is able to take into account its context of execution (i.e. context-aware). We place ourselves in the field of building construction. To build an application provides to each actor the whole of the services which it is likely to use within the framework of its activities concerns the challenge. Indeed, the actors move continuously in the building, the use of mobile entities of small size (e.g telephone, smartphone, pda, tabletPC, etc) is essential. However, being given weak characteristics (e.g capacity storage, limited autonomy, capacity of calculation, etc) of this type of apparatus, it is impossible to charge on each machine the services provided by our application. Moreover, diversity trades present in the framework of a building site imposes, for reasons of safety and comfort of use, only the services related to the trade are accessible by the actors concerned. Other parameters as the period of use must be taken into account. Indeed, at the same time as the building site evolves, the needs of the actors change. Thus, the application must be able to take into account all these elements in order to propose to each user the best possible services choice according to the environment of execution.

Thus, we propose to implement mechanisms allowing the ubiquitous business application of building site to take into account of its context of execution. This taking into account of the context can be resulted by the adaptation or the configuration of the application so as to provide to each actor a whole of judiciously selected services or in the proposal or the execution automatically for services appropriate to the context. An application answering these criteria should have architecture in layer. The first part refers to the material to support the application and to allow him to acquire all information of which it needs in order to adapt to its context. The second layer relates to modeling, the analysis and the treatment of the context which can be acquired via hardware or software sensor. The last layer contains the application

The application of building site management is to be carried out in a ubiquitously environment and mobile, the users must be able to reach the information provided by an infrastructure distributed (i.e services suggested by the application) without taking account of their site. The handled resources can be connected and disconnected constantly (i.e the user can leave constantly the zone of cover the building site, stop of the material, etc). Moreover, the quality of the network as well as the design features of the elements which make it up can also vary during their use.

2.1. Actors

In a building site, there are a great number of interveners whom we can classify according to their sector of intervention. There are four main categories of trade: the first relates to the phases of preparation and follow-up work (e.g architect, head clerk of studies, draughtsman, geometrician, engineer of studies in control of work, project superintendent, etc), the second category is centered on the carcass work heavy castings (e.g site foreman carcass work heavy castings, carpenter, mason, welder, mason stone, etc), the third category gathers all the trade of the second work (e.g tile-layer, heating specialist, roofer, electrician, carpenter, plasterer, etc) and finally the last category relates to the completions and the maintenance of the building (e.g. domitian, mirror, painter, moquettist, ascensorist, etc).

We also observe hierarchies of each trade. For example, the trades of preparations of work, we can evoke the following hierarchy: person in charge, principal assistant, assistants, etc Concerning the trades related to the carcass work heavy castings, the second work like the completion and maintenance, we speaks rather than a hierarchy owner, team leader, workmen, etc

2.2. Services

The objective of the application which we wish to implement consists in providing to each actor the whole of the services which can be useful for them at the time of the realization of work within the building site.

The services which the application proposes to provide to the various actors can be divided into two categories Research an object or a person in the building site:

- The whole of services independent of the trades (i.e services accessible by all the trades constantly)
    - Research an object or a person in the building site
    - call a principal in the building site
    - Show their positions in the building site
    - Consult the elements of context (context-aware passive)
    - Consult diary or program meetings
    - Consult general information (weather, date and time etc)
- The whole of specific services to the trades and the hierarchy in the trades For example, for the architect, the application must provide him the whole of the services as follow:
    - Budgetary management
    - Management of the contacts
    - Management of the planning of work
    - Graphic modeling of the whole of the project
    - Management of the authorizations
    - Management of the works and the detailed plans of all the structural components (Synchronization of the services provided by the other speakers, etc)
    - Management of the coordination of work's execution in conformity with its plans, their deadline and schedules of conditions.

2.3. Architecture of material

The objects using the composition of the network on which will be carried out the application are communicating objects (i.e fixed entities), the points of mobile entrance and final of the passive objects.

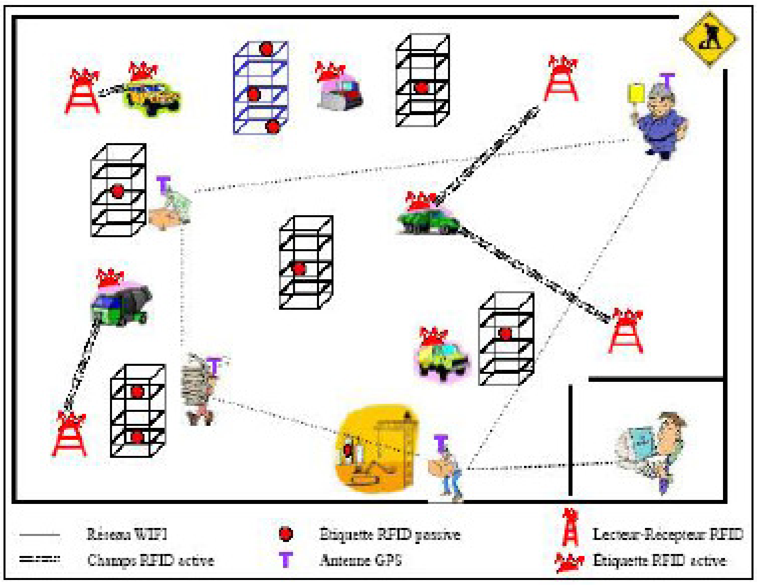


FIG.2- Architecture of building site

1. Communicating objects (i.e fixed entities): they are used as relay between information (i.e nodes) and station of data storage.

   a) Servers

   The servers are used like application supports. They interact with the various mobile entities associated with the agents via a network without wire (i.e generally Wifi). They make it possible to store the data (e.g building site plans, employs lists, available materials, diaries, etc) and are also intended to receive safeguards. They make it possible to conserve a history of building site (i.e work completed, conditions, etc). The servers are never disconnected.

   b) Wifi terminals

   The Wifi terminals are used as connector between various entities of the networks. They cover the totality of the building site and can be used as the users' localization (i.e geolocalized wifi).

   c) Weather Station equipped with various sensors allowed the data acquisition in real time in the atmospheric conditions (e.g thermometer, anemometer, pluviometer, barometer, etc)

   The weather station is positioned at a strategic place of the building site. It must be connected to a server which is able to interpret these data in order to provide to the other entities of the network a whole of services related to metrology. In addition, it has to conserver a history of the data thus recovered.

2. Mobile entrance points: they are generally used like tools (i.e footbridges) permit to access a network generally without wire (e.g. GSM, GPRS, WAP, Wifi, etc) in order to increase the

mobility of the user. The last one can constantly connect and disconnect with the network. Among these objects, we can quote the RFID label, the mobile phone and other smartphones, the PDA, the tabletPC,

3. Passive objects: these elements regroup the whole of the tools intended to acquire and analyze elements of the context. For example, a weather station equipped with various sensors allowed the data acquisition in real time on the atmospheric conditions is integrated into the network. It is positioned at a strategic place of the building site. It must be connected to a server which is able to interpret these data in order to provide to the other entities of the network a whole of services related to the metrology. In addition, it has to conserver a history of the data thus recovered.

The fixed entities are connected by means of telegraphic network and are laid within the building site according to the needs of contrary mobile entities which do not have a position fixes and which communicate only by network wireless telegraphy (i.e. Wifi, GSM/GPRS, Bluetooth, infra-red, etc). Thus, a mobile entity must be able to communicate with the fixed entities but also with the other mobile entities present on the building site.

Each agent is equipped with a mobile entity according to its trade and level in the hierarchy. For example, an architect will be equipped with TabletPC or a portable because the size of the screen appears among the largest in the category of the variable components (i.e in order to facilitate the visualization of charts).

Each object (e.g building, material, etc) is equipped with a chip RFID which makes it possible to identify and locate it in the building site.

Only the building site must be covered by the application, the use of geolocalized wifi terminals is essential to manage the access to the application.

2.4. Communication Element: WIFI

WI-FI is a brand originally licensed by the Wi-Fi Alliance to describe the underlying technology of wireless local area networks (WLAN) based on the IEEE 802.11 specifications. Wi-Fi is now so pervasive, and the term so generic, that the brand is no longer protected.

WIFI Allows LANs to be deployed without cabling, potentially reducing the costs of network deployment and expansion. Spaces where cables cannot be run, such as outdoor areas and historical buildings, can host wireless LANs.

Wi-Fi networks support roaming, in which a mobile client station such as a laptop computer can move from one access point to another as the user moves around a building or area.

2.5. Modeling application

To build our application, we use the technology of the software components because it enables us to handle high level software units and gain the reutilisability. Moreover, all the stages of the cycle of an application life (i.e design, deployment, execution, etc) are taken into account by the model what makes it possible to facilitate its adaptation. Our application of building site ubiquities is thus an assembly of components which can be distributed on the whole of the network. Each component provides a whole of services and can require the function of the services provided by other components. The functional components (i.e trade) of our application are as follows:

- A general services component

This component provides the independent services of the trade. It has to be charged on all the mobile entities connected to the network.

- Specific Components of each trade

Each trade is associated with a software component which will provide the whole of the services corresponding to it. These components could be charged on the mobile entities according to the profile of their user.

- A weather component

These services are regrouped in a software component charged on the server connected to the weather station.

- Components of safeguard and management of histories

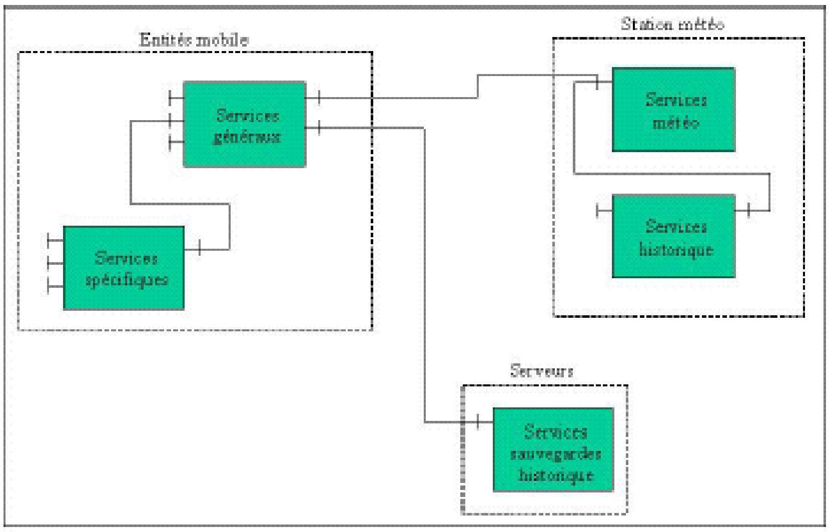


FIG.3- model of application

2.6. Deploying the application

When a mobile entity is connected to the network of the building site, it must be identified with a server and transmit the profile of its user to him. The server then will select the suitable components to transmit them to the mobile entity. The recovered components are charged.

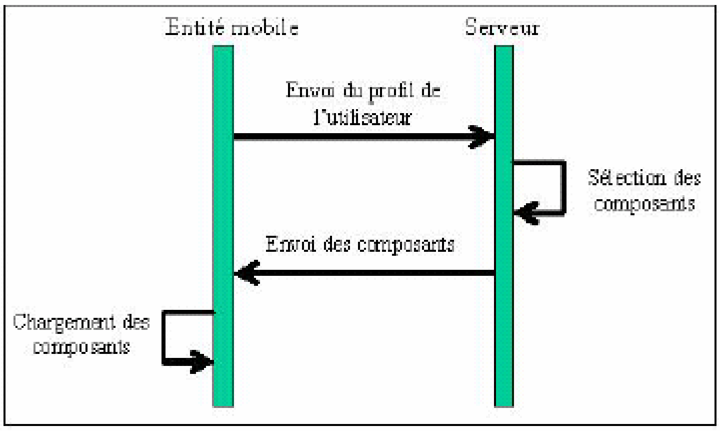


FIG.4- Deploying the application

3. Context of building site

3.1. Defining context

In our application, we define the context as the whole of the measurable or identifiable elements of the application and being able to influence the behavior of the application. The execution context of the application is strongly related to the presence of sensor making it possible to carry out its acquisition like the elements provided by the user (i.e user's profile). In our case, the elements of the context which we identified are as follows:

- User's profile
  - The trade: user's profile
  - the position in the hierarchy: user's profile

In the building site, we can consult people's situation according to their profiles. If someone is absent, we can replace his work according to the hierarchy.

- Close environment (presence of the other actors or objects on the building site): emission of a signal (wifi,GSM,…)
- Localization: GPS, wifi, RFID or GSM

We can know people's position, particularly their absent by this context.

- Moment: internal clock
- Atmospheric Conditions:
  - Temperature(thermometer)
  - Pressure (barometer)
  - force of the wind (anemometer)

- direction of the wind (wind vane)
- content water
- indication rain (pluviometer)

When the special atmospheric conditions happen, we can send alarms to person.

3.2. Why we need context in the application

The taking into account of the context by the application is very important, particularly in unbiquitous computing. The objectives of the applications "context-aware" are to enrich and facilitate the man-machine interactions while making the applications sensitive to the environment and to anticipate the actions of the user (i.e. what it tries to do and what it perhaps will do). For example, for a portable telephone, if the user is in a not very luminous environment, the screen must light automatically before the user did not choose to start this operation.

Indeed, the taking into account of the user's needs is done by the analysis of the material resources as well as environment of the user. The problem remains the context and the user's needs evolve permanently. The application must be able to manage that.

3.3. How the context is carried out?

The applications "context-aware" are applications equipped with tools which make it possible to take into account their context of execution. Their process of adjustment comprises four stages:

- Collection of information relating to the context

The acquisition of contextual information is done by using simple sensors (e.g thermometer, anemometer, etc), by resource administrators (e.g. Niveau of the battery, etc) or even more sophisticated tool resource administrators for supervision.

- Analyze the information

The analysis of the collected data during the preceding stage must make it possible to extract the relevant context (i.e context which affects the behavior of the application during sound deployment or during its execution).

- Making decision

According to the collected indications during the stage of analysis, the adapter must establish a strategy of adaptation. It can consist in for example choosing the implementation of the elements which compose it, the structure of the application or the establishment machines of its components best adapt to the situation.

- Action

The last stage of the process consists in implementing the decisions taken in the preceding stage. The actions to be realized can be varied: modification of parameters (i.e reconfiguration), adaptation of the application (e.g handling the code, transfer of code, replacement of services, redirection of messages, etc) evolution of application (e.g addition/suppression of functionalities, etc), presentation of the data to the user, etc

3.4. Acquisition of context

Each mobile entity are equipped with sensors which can be software (i.e indicator level of battery, indicator of bandwidth of the network, etc) or material (e.g sensor of noise, sensor of luminausity, etc) and making it possible to acquirer knowledge on the context close to the user. The position of the user within the building site is given differently if the person is inside a building or outside. Outside, a sensor GPS used to determine the position. Interior, it is necessary to use other technique such as the geolocalized wifi or chips RFID. The atmospheric conditions can be detected by thermometer, barometer and anemometer etc.

3.5. Modeling context

Context Toolkit provides an environment making it possible to develop context-aware applications by use of widgets which encapsulate the sensors. These elements can be organized as a hierarchical architecture which supports basic processes like aggregation of information. In fact, there are interfaces which make it possible to hide the application, the operations of acquisition of contextual information.

A widget of context acquires all information necessary for each attribute. For each entity of context, an aggregator collects information necessary starting from the widgets. The applications can then consult the entities of context created. Interpreters can be used to facilitate the application of the context. They are charged to interpret the information collected on the context.

4. Realized services

4.1. Service descriptions

4.1.1. Researching person

In a building site, there are a great number of interveners whom we can classify according to their sector of intervention. Sometimes we have to know their profile which contains their name, trade. Everyday we have to know whether they are present. If they are present, we have to know their real-time localization. If they are absent, who can replace them to do their work. We can see the function in the use case diagram as follow:

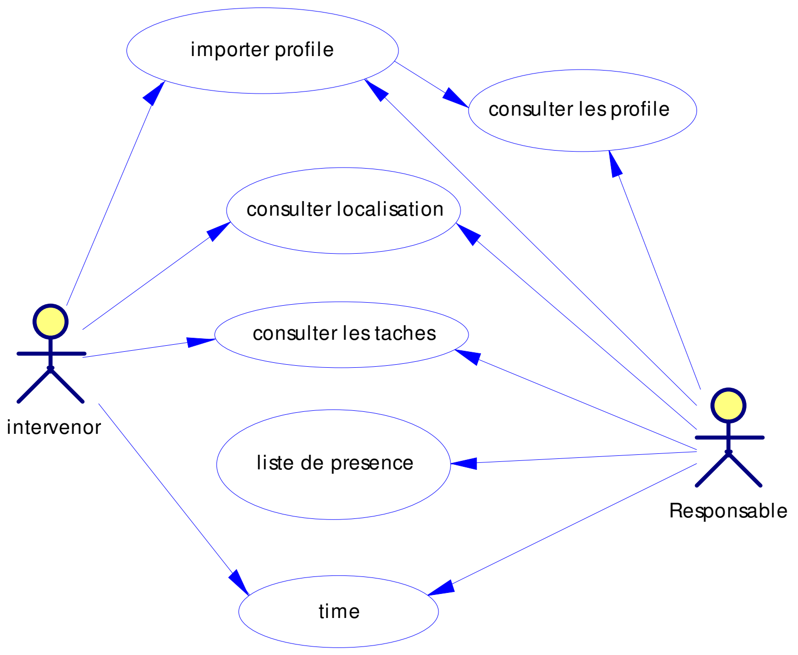


FIG.5- use case of profile

4.1.2. Alarmes

In the building site, there are many types of context. According to the different context, we can develop different alarms. We can divide the alarms into two types:

- Information alarms

When a particular situation happens, we can inform people in the building site immediately. For example when it's rainy, the server can send an alarm to all people's OMC automatically.

- Reactive alarms

This alarms mean that people who receive the alarm have to do some task corresponds to the alarm. For example the temperature is more than 35 degrees, it's rainy or the wind is too strong; we have to send an alarm to a person or a group of person who should close the window. We can see an example in the sequence diagram as follow:

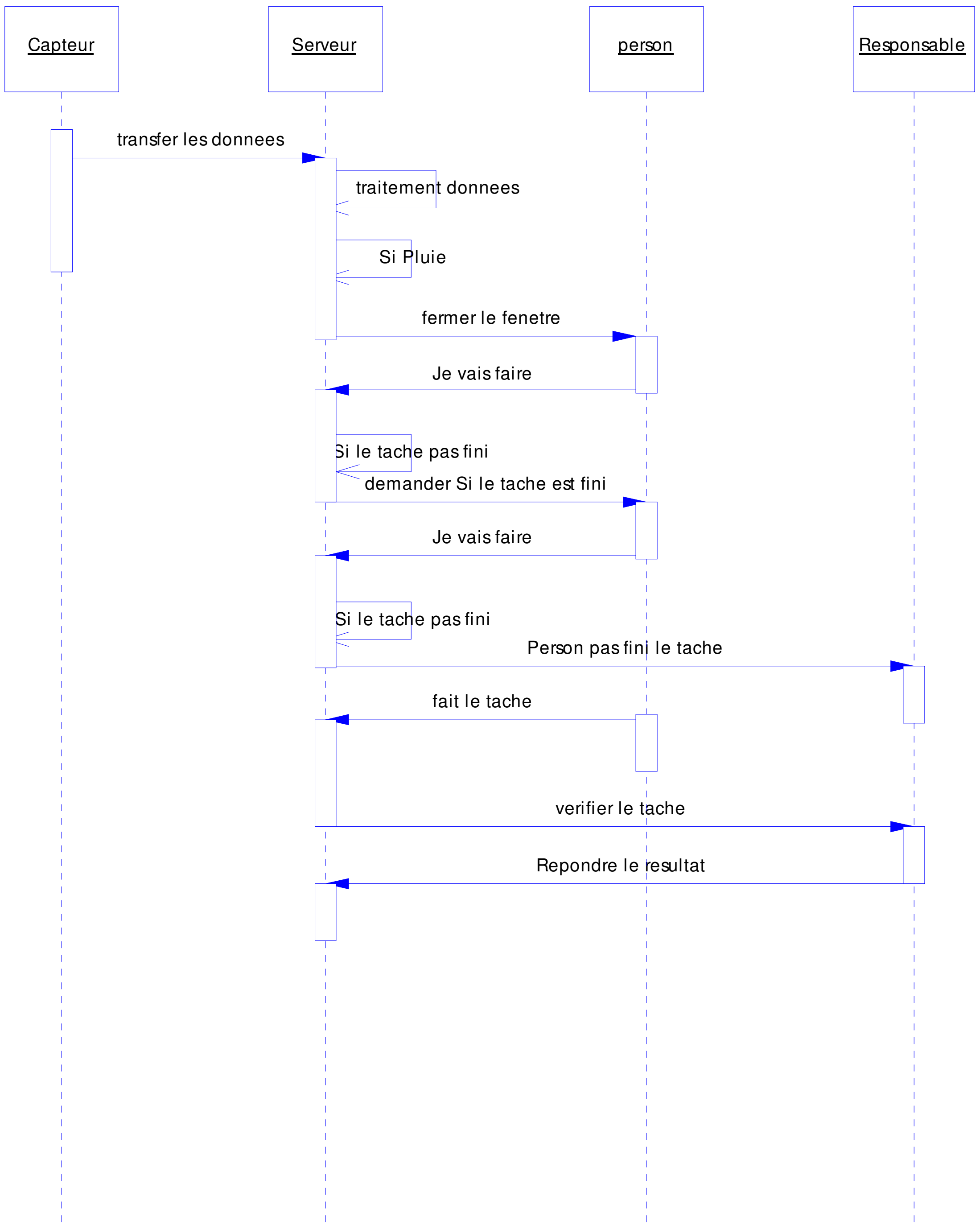


FIG.6- Sequence diagram of alarms

4.1.3. Organizing a meeting

In the building site, many meetings are often organized (For example, planning the future work). According to the way of organizing, the meetings can be divided into two types: preorder meeting and temporary meeting.

- Preorder meeting

It means that the meeting is organized at certain day of every week or every month for, by example, concluding works of every month. In order to realizing it, we can send a message to all the members to notify them at certain day automatically.

- Temporary meeting

Sometimes special situations will happen in the building site. Some people have the right to organize a temporary meeting. For example, if current works have been finished, the principal has to organize a meeting to plan the future works. These people have an interface for demanding the organization of a meeting. They can select the persons who take part in the meeting and period of the meeting. An alarm will be sent to everyone selected. The persons receiving the alarm have several choices: confirmation of presence, I can't be present. According to the choice of the person having received alarm another message is sent to OMC of the person having required the organization of the meeting. The various diaries concerned are updated. There are replacement rules of absent persons and rules of delegation.

We can see the precise in the sequence diagram and activity diagram.

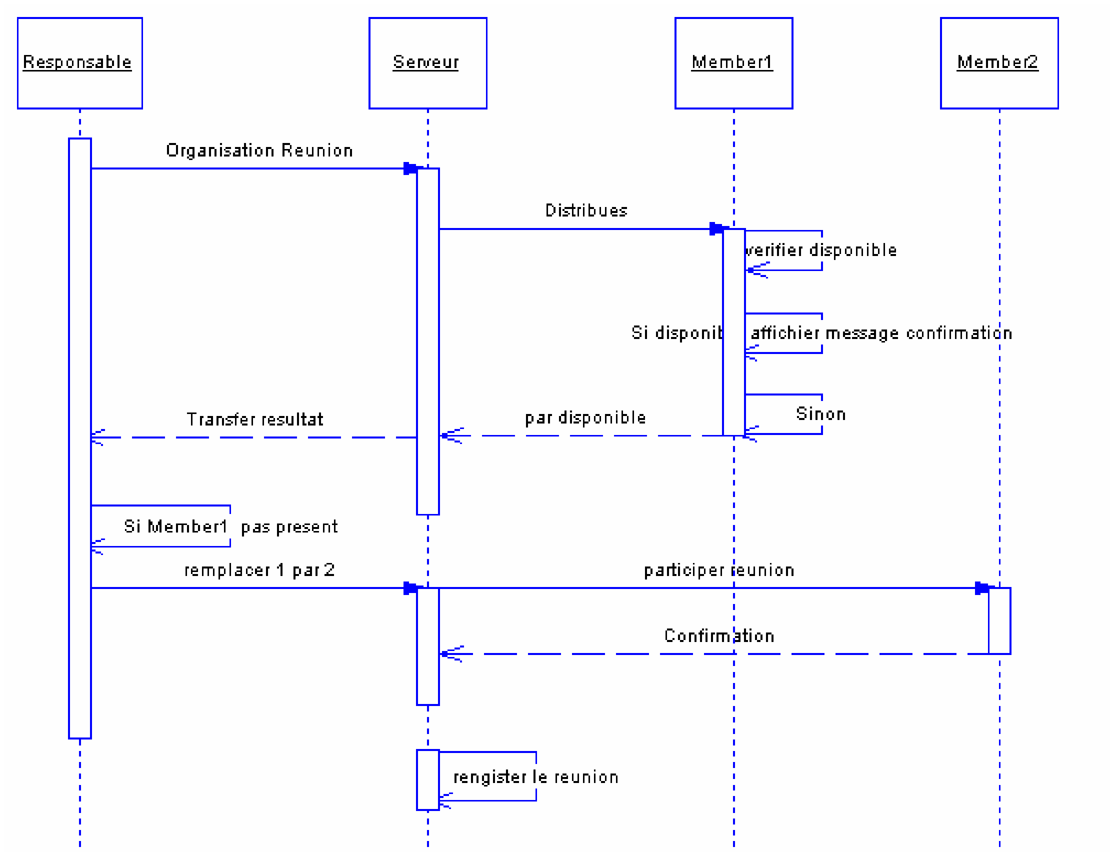


FIG.7- Sequence diagram of meeting

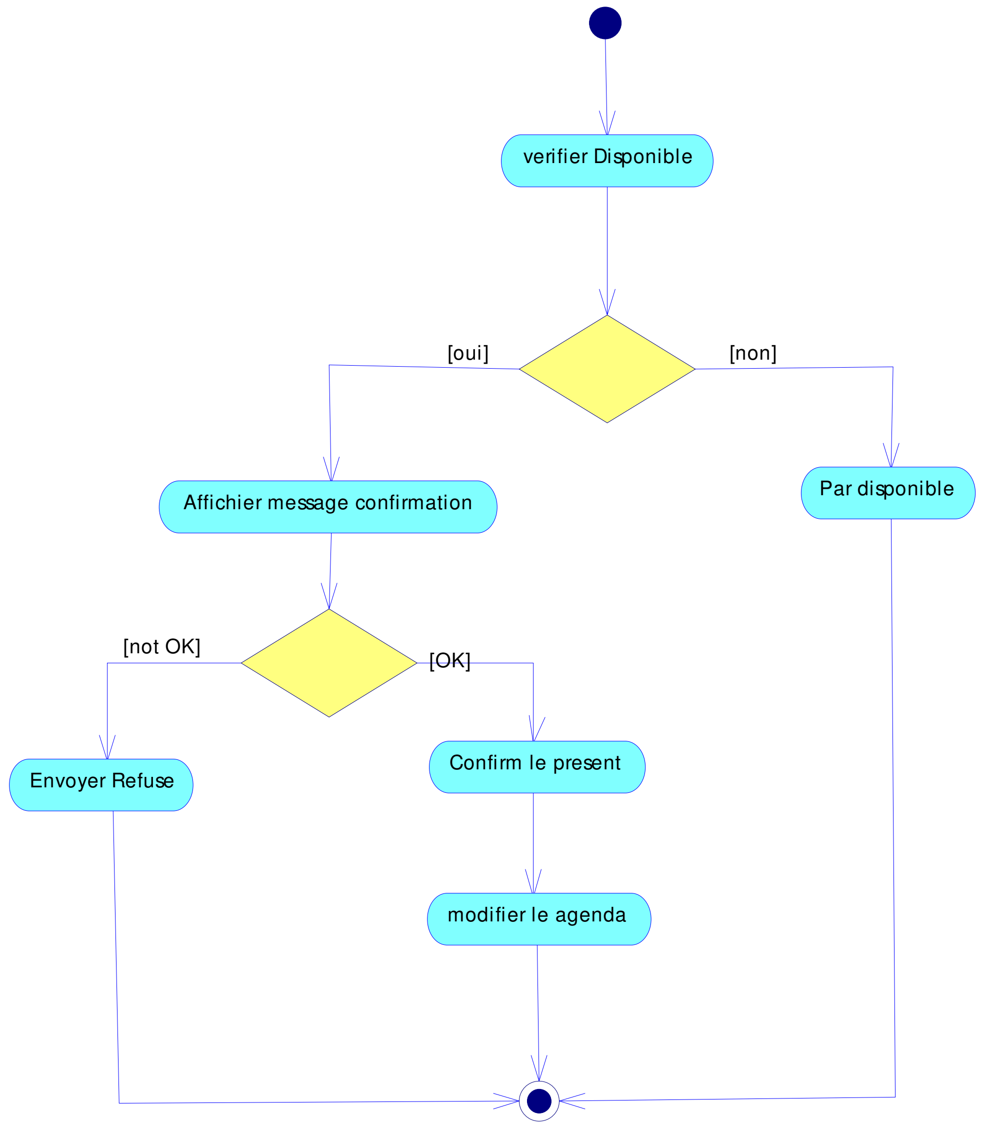


FIG.8- Activity diagram of meeting

4.1.4. Chat

Sometimes people should communicate with each other in the building site. So the development of chat service is necessary. When someone wants to chat with other people, he can send an alarm to their OMC to notify them. If the people he wants to chat are far from him, they should chat online via their OMC. Conversely he can demand them to go to his office and chat face to face. We can see the function in the use case diagram as follow:

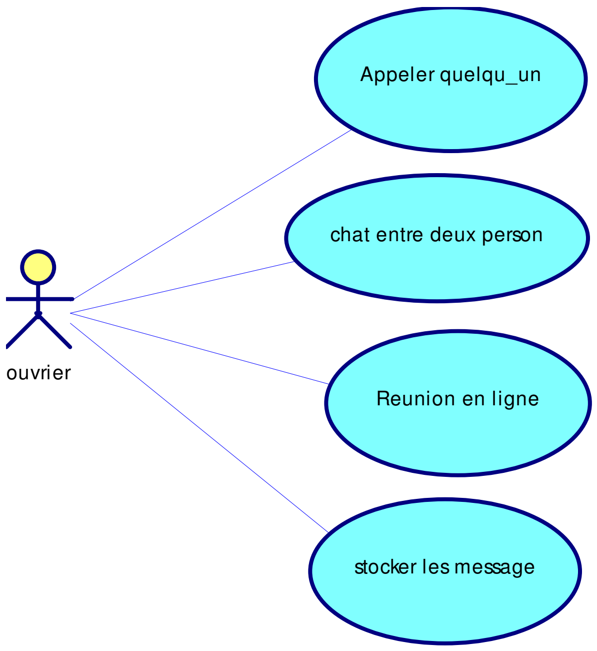


FIG.9- use case diagram of chat

4.2. Implementation

4.2.1. Using database in Java

In the services, the widget acquires the information from sensors. And then the server collects the information from widget and saves in the database MySQL. The application can acquire the information from MySQL and show on an interface. But For the information saved on the OMC, we use file XML, because of the limitation of the OMCs' memory.

4.2.1.1. MySQL

In this system we have to install a database to save data on the server. I choose MySQL. MySQL is the database server more used in the world. Its software architecture makes it extremely fast and easy to personalize. The principal advantages of MySQL are its speed, its robustness and its facility of use and administration. Another major advantage of MySQL is its very complete and well built documentation.

We'll show you the structure of the database in the figure follow.

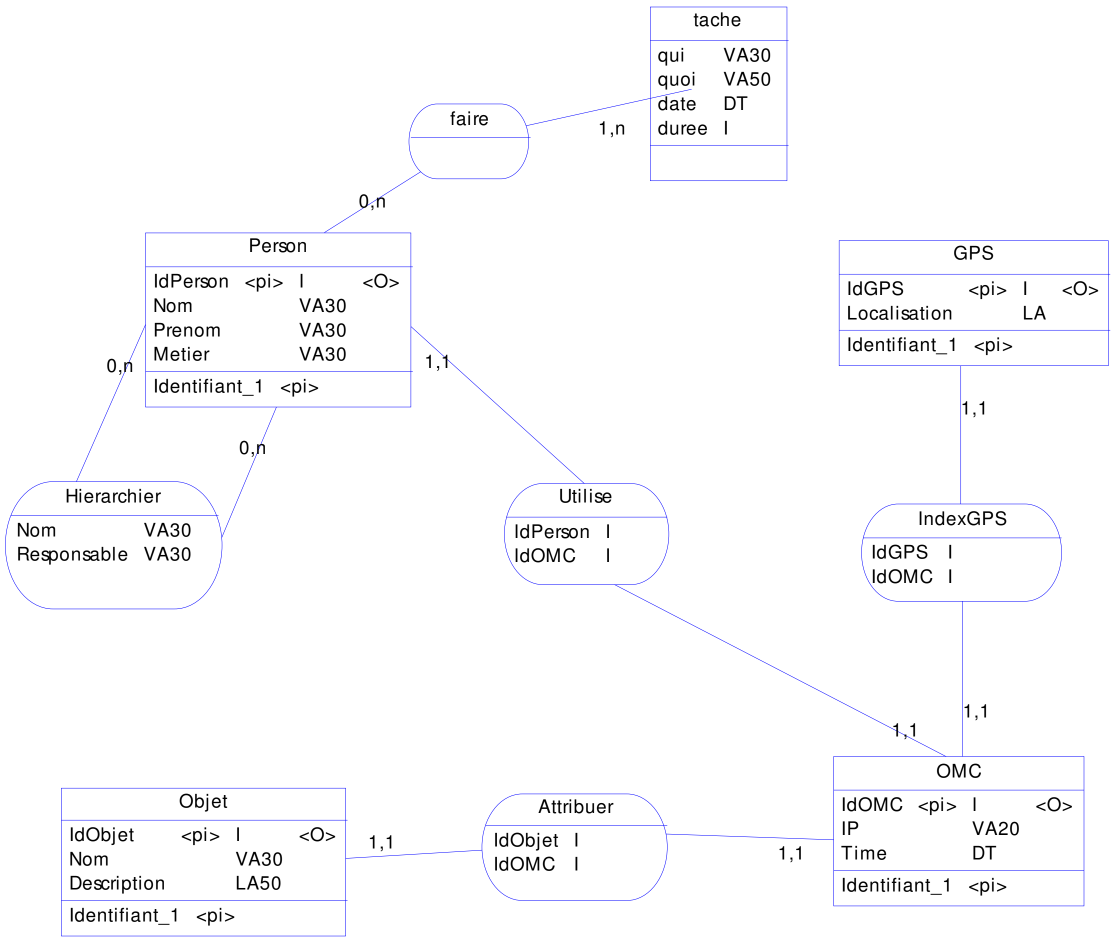


FIG.10- MCD of Database

- Table person: The profile of person
- Table objet: The object in the building site
- Table OMC: The mobile equipment (PDA, smartphone etc)
- Table GPS: GPS for localization
- Table tache: The list of tasks in the building site

4.2.1.2. XML

In the program Java, we can read and write a XML file by adding External jar Crimson or using Class SAX. But the OMC only support Java 1.3. There isn't Class SAX in Java 1.3. We can't use Crimson too on the OMC. So we can only read and write a file XML like text file.

In the services, we can create files XML to save data on the OMC. For example, in the service Alarms, We can create a file XML to save the data for the task. It has to contain the task's identification, description, reception date and period date etc.

4.2.2. Communicating between OMC and server

The application will send messages to people. So we have to realize the communication between application and people's OMC. Because the clients are mobile, we will use WiFi in the communication.

4.2.2.1. Communication types

According to the different functions of communication, we use two types of communications: when OMCs send messages to application, we use Socket. When the application sends messages to OMCs, we use MulticastSocket.

4.2.2.1.1. Socket

Sometimes your programs require lower-level network communication, for example, when you want to write a client-server application.

In client-server applications, the server provides some service, such as processing database queries or sending out current stock prices. The client uses the service provided by the server, either displaying database query results to the user or making stock purchase recommendations to an investor. The communication that occurs between the client and the server must be reliable. That is, no data can be dropped and it must arrive on the client side in the same order in which the server sent it.

TCP provides a reliable, point-to-point communication channel that client-server applications on the Internet use to communicate with each other. To communicate over TCP, a client program and a server program establish a connection to one another. Each program binds a socket to its end of the connection. To communicate, the client and the server each reads from and writes to the socket bound to the connection.

- **What Is a Socket?**

A socket is one end-point of a two-way communication link between two programs running on the network. Socket classes are used to represent the connection between a client program and a server program. The java.net package provides two classes--Socket and ServerSocket--that implement the client side of the connection and the server side of the connection, respectively.

a) Client

The client typically creates a Socket, gets I/O streams from it, and then does consecutive writes and reads.

```
socket = new Socket("169.254.215.6", 1131);

in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

out = new PrintWriter(socket.getOutputStream(),true);
```

b) Server

A server uses a ServerSocket to bind to a port and listen on it. When a new client request is accepted, it returns a Socket which is connected to the client. The server uses this to talk to the client, typically reading requests and responding to them.

```
ss = new ServerSocket(1121);

socket = ss.accept();

in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

out = new PrintWriter(socket.getOutputStream(),true);
```

4.2.2.1.2. MulticastSocket

The multicast datagram socket class is useful for sending and receiving IP multicast packets. A MulticastSocket is a (UDP) DatagramSocket, with additional capabilities for joining "groups" of other multicast hosts on the internet.

A multicast group is specified by a class D IP address and by a standard UDP port number. Class D IP addresses are in the range 224.0.0.0 to 239.255.255.255, inclusive. The address 224.0.0.0 is reserved and should not be used.

One would join a multicast group by first creating a MulticastSocket with the desired port, then invoking the joinGroup(InetAddress groupAddr) method.

When one sends a message to a multicast group, all subscribing recipients to that host and port receive the message (within the time-to-live range of the packet, see below). The socket needn't be a member of the multicast group to send messages to it.

When a socket subscribes to a multicast group/port, it receives datagrams sent by other hosts to the group/port, as do all other members of the group and port. A socket relinquishes membership in a group by the leaveGroup(InetAddress addr) method. Multiple MulticastSocket's may subscribe to a multicast group and port concurrently, and they will all receive group datagrams.

4.2.3. Expert System

An expert system also known as a knowledge based system, is a computer program that contains some of the subject-specific knowledge of one or more human experts. This class of program was first developed by researchers in artificial intelligence during the 1960s and 1970s and applied commercially throughout the 1980s. The most common form of expert systems is a program made up of a set of rules that analyze information (usually supplied by the user of the system) about a specific class of problems, as well as providing analysis of the problem(s), and, depending upon their design, recommend a course of user action in order to implement corrections. It is a system that utilizes reasoning capabilities to reach conclusions.

4.2.3.1. Why we use Expert System?

Expert systmes exercise information technology to acquire and utilize human expertise. It can be beneficial for organizations that have clear objectives, rules and procedures. Expert systems can:

- Provide consistent answers for repetitive decisions, processes and tasks
- Hold and maintain significant levels of information
- Reduce employee training costs
- Centralize the decision making process
- Create efficiencies and reduce time needed to solve problems
- Combine multiple human expert intelligences
- Reduce the amount of human errors
- Give strategic and comparative advantages creating entry barriers to competitors
- Review transactions that human experts may overlook

4.2.3.2. Jess

Jess is a rule engine and scripting environment written entirely in Sun's Java^{TM} language by Ernest Friedman-Hill at Sandia National Laboratories in Livermore, CA. Using Jess, you can build Java software that has the capacity to "reason" using knowledge you supply in the form of declarative rules. Jess is small, light, and one of the fastest rule engines available. Its powerful scripting language gives you access to all of Java's APIs.

Jess uses an enhanced version of the Rete algorithm to process rules. Rete is a very efficient mechanism for solving the difficult many-to-many matching problem. Jess has many unique features including backwards chaining and working memory queries, and of course Jess can directly manipulate and reason about Java objects. Jess is also a powerful Java scripting environment, from which you can create Java objects, call Java methods, and implement Java interfaces without compiling any Java code.

4.2.4. RMI

Remote method invocation allows applications to call object methods located remotely, sharing resources and processing load across systems. Unlike other systems for remote execution which require that only simple data types or defined structures be passed to and from methods, RMI allows any Java object type to be used - even if the client or server has never encountered it before. RMI allows both client and server to dynamically load new object types as required.

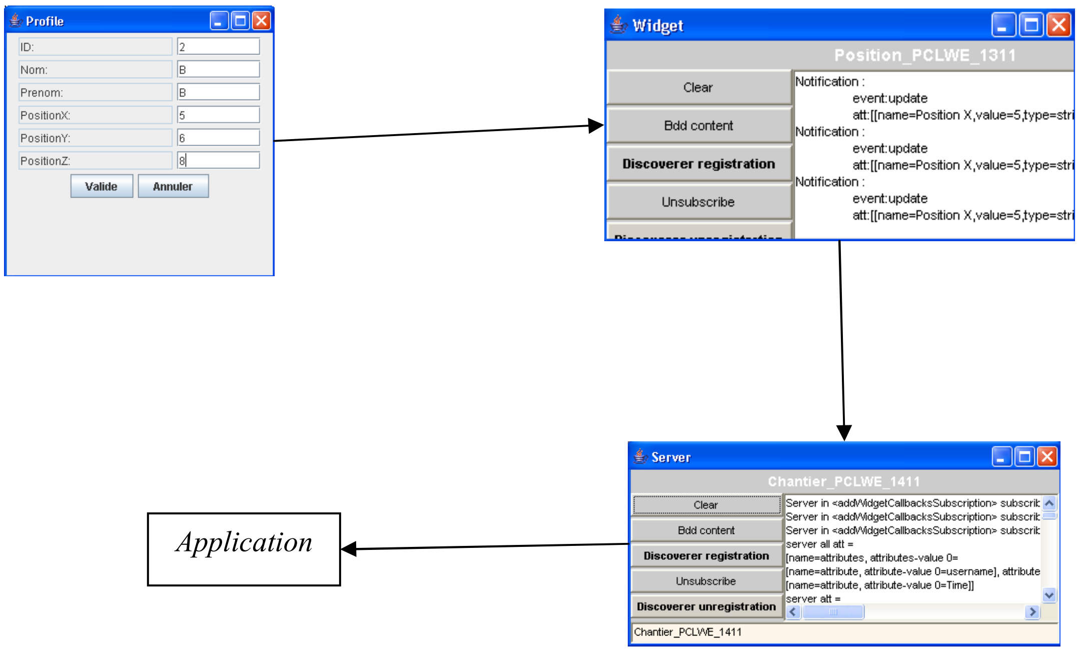
The reason why we use RMI in our project is the limited resource of OMC. If the services occupy too many memories, we can't install all the services on the OMC. But we can use RMI to call the services located remotely.

4.3. Manuel of interface

4.3.1. Researching person

- **Simulation of sensors**

The profile interface can simulate the people's profile and their localizations. We can input the name and localization and then press the button "valide". The information will be sent to database MySQL via Widget and Server.



ID:	2
Nom:	B
Prenom:	B
PositionX:	5
PositionY:	6
PositionZ:	8

```
Notification :
        event:update
        att:[[name=Position X,value=5,type=stri
Notification :
        event:update
        att:[[name=Position X,value=5,type=stri
Notification :
        event:update
        att:[[name=Position X,value=5,type=stri
```

```
Server in <addWidgetCallbacksSubscription> subscrib
Server in <addWidgetCallbacksSubscription> subscrib
Server in <addWidgetCallbacksSubscription> subscrib
server all att =
[name=attributes, attributes-value 0=
[name=attribute, attribute-value 0=username], attribute
[name=attribute, attribute-value 0=Time]]
server att =
```

Chantier_PCLWE_1411

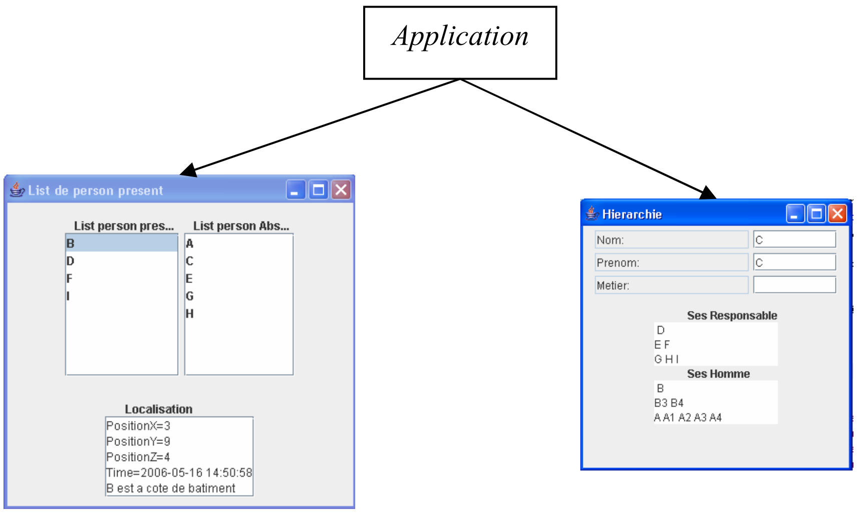
FIG.11- simulation of sensors

- **Application**

The present interface lists all the people who are present in the building site and who are absent. When we choose a people in the list, the text area will show the last absolute position, relative position and time.

In this model, if GPS's coordinate of person is out of the building site, which means this person is absent. If the distance between person and an object stable is less than some meter, which means this person is near by this object.

The hierarchic interface shows the hierarchic relating to this person: his underling and his boss.



```
List person pres...   List person Abs...
B                     A
D                     C
F                     E
I                     G
                      H

Localisation
PositionX=3
PositionY=9
PositionZ=4
Time=2006-05-16 14:50:58
B est a cote de batiment
```

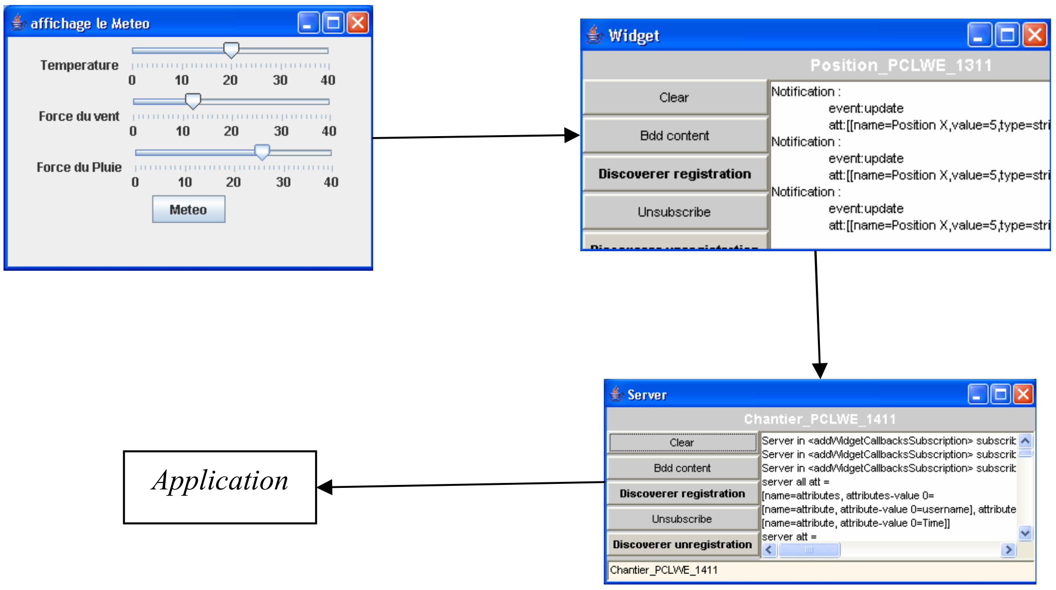
```
Nom:     C
Prenom:  C
Metier:

Ses Responsable
D
E F
G H I
Ses Homme
B
B3 B4
A A1 A2 A3 A4
```

FIG.12- application of researching

4.3.2. Alarms

- **Simulation of sensors**

We simulate the change of temperature, rain and wind by sliding the buttons. When we change the data, it will be acquired by Widget and server. For example, if the temperature is more than 30, the server will send an alarm to the principal of weather's OMC.

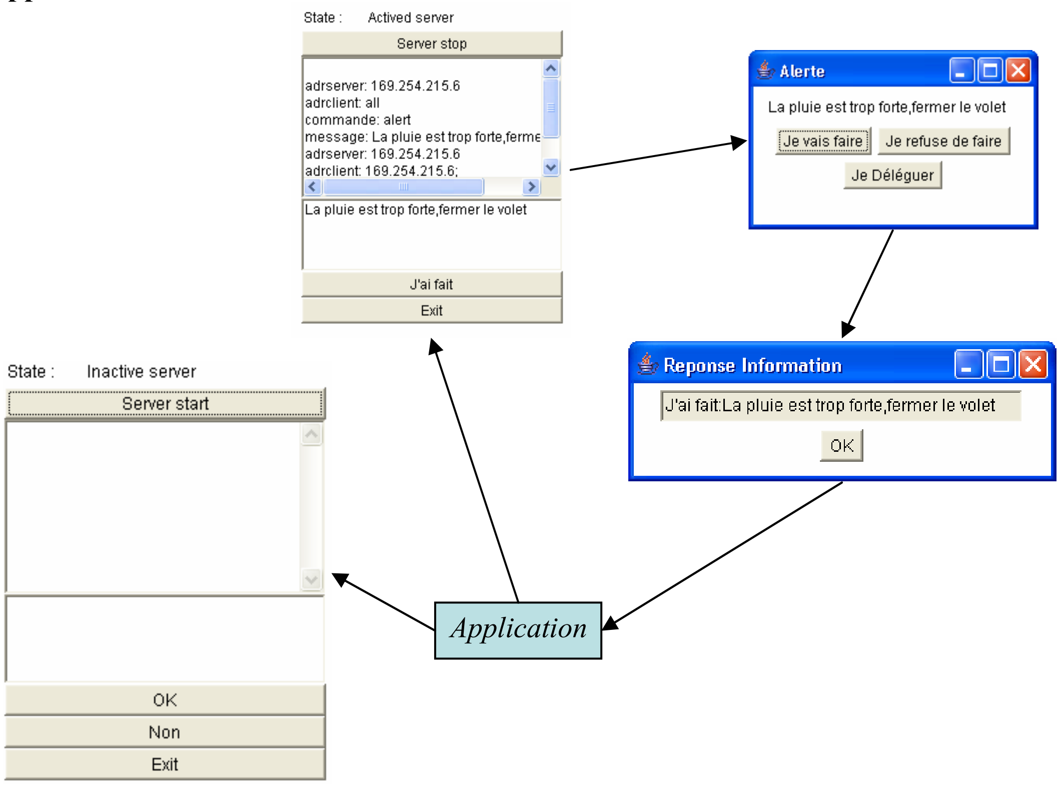


```
Notification :
        event:update
        att:[[name=Position X,value=5,type=stri
Notification :
        event:update
        att:[[name=Position X,value=5,type=stri
Notification :
        event:update
        att:[[name=Position X,value=5,type=stri
```

```
Server in <addWidgetCallbacksSubscription> subscrib
Server in <addWidgetCallbacksSubscription> subscrib
Server in <addWidgetCallbacksSubscription> subscrib
server all att =
[name=attributes, attributes-value 0=
[name=attribute, attribute-value 0=username], attribute
[name=attribute, attribute-value 0=Time]]
server att =
Chantier_PCLWE_1411
```

FIG.13- simulation of sensors

- **Application**



```
State :     Actived server
adrserver: 169.254.215.6
adrclient: all
commande: alert
message: La pluie est trop forte,ferme
adrserver: 169.254.215.6
adrclient: 169.254.215.6;
La pluie est trop forte,fermer le volet
```

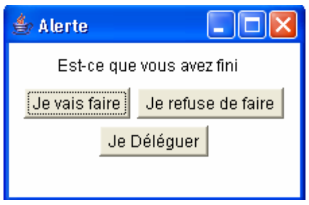
```
State :     Inactive server
```

Alerte: La pluie est trop forte,fermer le volet

Reponse Information: J'ai fait:La pluie est trop forte,fermer le volet

FIG.14- Application of alarms

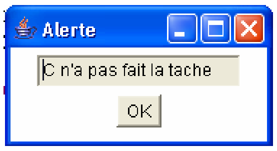
On the OMC, We can see the information received from application and the list of tasks the person has to do. When the application sends an alarm to the OMC, it will spring an alarm window. The person can choose different action. For example, if he chooses "je vais faire", the task will add into the list of task. At the same time, the choice is transmitted to the application, according to the choice made by the person. Then application carries out a policy of recalls. When the task can still be realized without problem urgently, send message following all X period.



Alerte: Est-ce que vous avez fini

FIG.15- Interface of resending

If the task isn't realized in this period, the application will send a special alarm to some person (principal of this person).



Alerte: C n'a pas fait la tache

FIG.16- Interface of negation

If the task has been done, the person can press “J’ai fait” to send a message to the application. Then the application sends a message to the person’s principal for verifying the task.

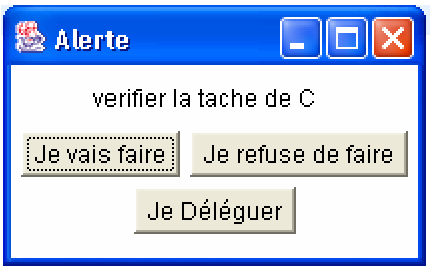
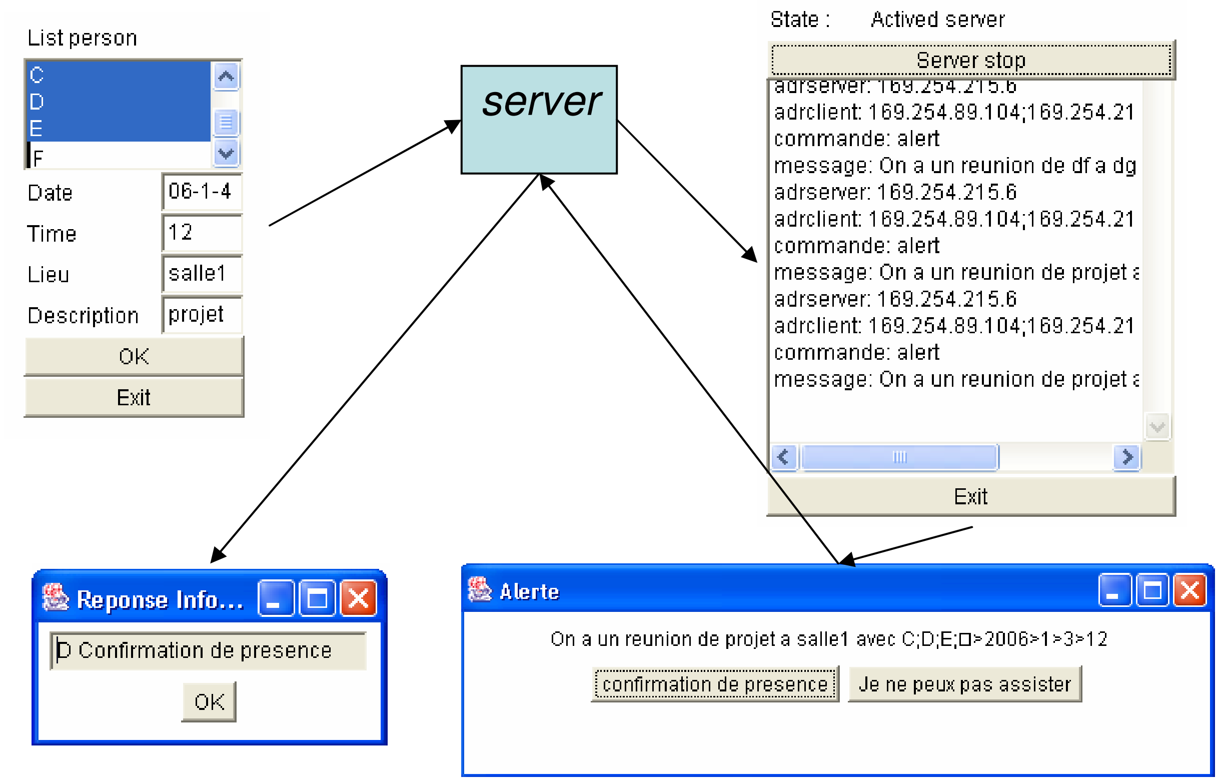


FIG.17- Interface of verification

On the principal’s OMC, there is an interface for answering the application. If he presses “OK”, it means that the task which the person has done is fine. If he presses “Non”, it means the task hasn’t been finished.

4.3.3. Organizing a meeting

There is an interface of meeting on the OMC of principal who can organize a meeting. He can choose the people who take part in the meeting in the list and input the day, time, place and content of the meeting which will send to everyone.



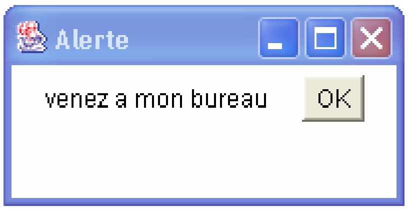
```
adrserver: 169.254.215.6
adrclient: 169.254.89.104;169.254.21
commande: alert
message: On a un reunion de df a dg
adrserver: 169.254.215.6
adrclient: 169.254.89.104;169.254.21
commande: alert
message: On a un reunion de projet a
adrserver: 169.254.215.6
adrclient: 169.254.89.104;169.254.21
commande: alert
message: On a un reunion de projet a
```

On a un reunion de projet a salle1 avec C;D;E;□>2006>1>3>12

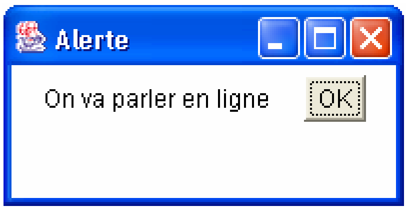
FIG.18- Interface of meeting

4.3.4. Chat

For the service chat, an interface which can install on the OMC is created. When the user wants to chat with other people, he can choose the people he wants to chat in the list ("A,B,C…" in the interface). He can notify them by press "chat". At the same time the people who are notified can receive an alarm. According to the distance between the chat persons, they will receive different alarms. If they are far from each other, they prefer to chat online via their OMC. Conversely they prefer to chat face to face in the office. We can see two alarms as follows:



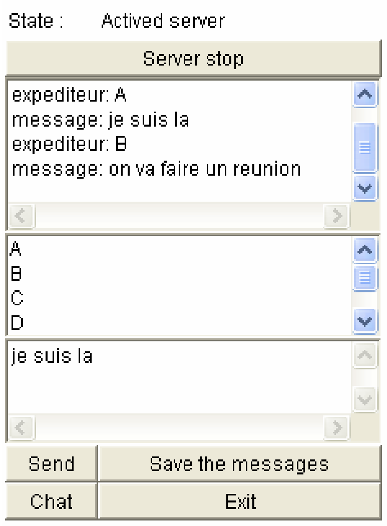
venez a mon bureau



On va parler en ligne

FIG.19-Alarmes of chat

If they chat online, they can input messages in the "TextField" and press "send" to send to each other. At last they can press "save the messages" to save the messages in a text file.



State : Actived server

expediteur: A
message: je suis la
expediteur: B
message: on va faire un reunion

A
B
C
D

je suis la

FIG.20- Interface of chat

Conclusion

In this report, we have studied the theory of ubiquitous computing and Context-Aware, and then we have proposed an establishment of scenario of ubiquitous application.

The objective of the work is to develop the approach of adaptation structural of a software component for an effective management of limited resources in the ubiquitous environments.

In this objective, first we described the context of work, which are ubiquitous computing and Context-Aware. It contained their definition and characteristic. And then we described a framework of Context-Aware and its application in the project.

In this context, the work we presented in the report necessary to be completed and extended is:

- Development a ubiquitous scenario around the management of limited resource.
- Self-adaptation structural of a software component to ensure an effective management of the resources.

Lastly, the ubiquitous applications, by their intrinsic characteristics concerning the limitations of resources can constitute a field where the structural adaptation can bring certain solutions.

References

1. Brown, P.J. The Stick-e Document: a Framework for Creating Context-Aware Applications. Electronic Publishing ’96 (1996) 259-272

2. Brown P. J., Bovey J. D., Chen X., “Context-aware applications: From the laboratory to the marketplace”, IEEE Personal Communications, 4(5):58–64, 1997.

3. Dey, A.K. Context-Aware Computing: The CyberDesk Project. AAAI 1998 Spring Symposium on Intelligent Environments, Technical Report SS-98-02 (1998) 51-54

4. Dey, A.K., Abowd, G.D., Wood, A. CyberDesk: A Framework for Providing Self-Integrating Context-Aware Services. Knowledge-Based Systems, 11 (1999) 3-13

5. Franklin, D., Flaschbart, J. All Gadget and No Representation Makes Jack a Dull Environment. AAAI 1998 Spring Symposium on Intelligent Environments, Technical Report SS-98-02 (1998) 155-160

6. Hull, R., Neaves, P., Bedford-Roberts, J. Towards Situated Computing. 1st International Symposium on Wearable Computers (1997) 146-153

7. Ryan, N., Pascoe, J., Morse, D. Enhanced Reality Fieldwork: the Context-Aware Archaeological Assistant. Gaffney,V., van Leusen, M., Exxon, S. (eds.) Computer Applications in Archaeology (1997)

8. Rodden, T., Cheverst, K., Davies, K. Dix, A.. Exploiting Context in HCI Design for Mobile Systems. Workshop on Human Computer Interaction with Mobile Devices (1998)

9. Schilit B. N., Theimer M. M., “Disseminating active map information to mobile hosts”, IEEE Network, 8(5):22–32, September/October 1994.

10. Ward, A., Jones, A., Hopper, A. A New Location Technique for the Active Office. IEEE Personal Communications 4(5) (1997) 42-47

11. Weiser M, The Computer for the 21st Century, Scientific American, 1991.