A Cross-Platform and Context-aware Middleware Application

Liao Zhen
Beijing University of Posts and Telecommunications,
Beijing, China
10212756@bupt.edu.cn

Zhao Jingling
School of Computer,
Beijing University of Post and Telecommunications,
Beijing, China
zhaojingling@bupt.edu.cn

***Abstract*—Nowadays, context-aware applications have become a hot research field. However, many context-aware applications don't fully connect to the characteristics of the Internet and the developed applications are limited by the platform, thus, the cross-platform deployment can't achieve. This paper presents a mechanism which can shield the difference among terminal operating systems and gather, store, and interpret the context information on the same smart terminal, based on the research achievements about the arithmetic of context-aware and the design of the cross-platform and context-aware middleware. To some extents, this frame can not only lessen the developers' workload and avoid duplication of effort, but also speed up the applications' development time and reduce the power consumption of mobile terminal.**

Keywords-context-aware; cross-platform; middleware; mobile terminal

I. INTRODUCTION

With the rapid development of communication technologies, the mobile devices permeate almost every facet of our lives. And in order to meet the users' needs, the mobile devices' functions are extended and develop to the intellectualized direction. Since Weiser[1] proposed the concept of the ubiquitous computing, context-aware gets enough attention and become the hot topic of the computer field, such as, the "SmartClassroom" project by Tsinghua University [2], the "Oxygen" project by MIT [3], the "DreamSpace" project by IBM [4], etc. Its system can use context information automatically. Besides, it provides services and computer resources. Meanwhile, mobile devices become the ideal platform of context-aware because of these feature.

However, intelligent applications of the mobile devices about context-aware are still lacked. In the early system, in order to achieve the specific application, the developers must participate in the whole process from building sensors' environment, collecting the context information to programming, and applied logic and context acquisition are tightly coupled, which reduces the system's reusability. In addition, these applications need collect context information by their own programs, especially when sensors are changed, the high level context also must modify accordingly. This case increases the cost of development and difficulty. The main reason of this phenomenon is lack of a middleware to facilitate the development of such applications.

Thus, this paper proposes a framework of the middleware to centralize context processing work so that reduce the time and difficulty of developing context-aware application. This middleware contributes to simplify the process of context-aware and provide relevant service information to the high level.

In the section Ⅱ, this paper introduces some relevant developing frameworks about context-aware. Then, the section Ⅲ introduces the work mechanism with middleware and explains every model in detail. In the end, the section Ⅳ proposed the application based on this mechanism.

II. RELATED WORK

With the development of the context-aware application, many researchers or scholars are proposed the framework of it and improve it from many aspects, such as, modeling, inferring and collision detection.

Context Broker Architecture is the system of context-aware based on intelligent agent and CoBrA [5] uses Context Brober to manage the smart space of the context. In this process, the ontology model contributes to express the meaning of the context, but, the computing is complex and it's unfit for the mobile platform or environment which has limited resources. When the Internet's environment is limited, the system's performance fails to maintain stability.

BeTelGeuse [6] is proposed by HIT, which supports the communication between the mobile devices and external sensors by Bluetooth. However, the existing sensors don't equip the module of Bluetooth and they communicate by Zigbee. Besides, BeTelGeuse needs every external sensor to own one resolver. Thus, the extendibility of the middleware is reduced and this communication is not suit for the dynamic environment.

Therefore, to the development framework of the context-aware application, the characteristic of the mobile environment, the ways of context modeling, context storing and so on need to be considered.

III. ARCHITECTURE FOR THE CROSS-PLATFORM AND CONTEXT-AWARE MIDDLEWARE

This section describes details of middleware. It locates on the top of the server's operate system and it can control the resources and network communication. The middleware connects two independent applications even if they have different interfaces. By this way, the application can be adopted on multi-platform.

Published by Atlantis Press, Paris, France.

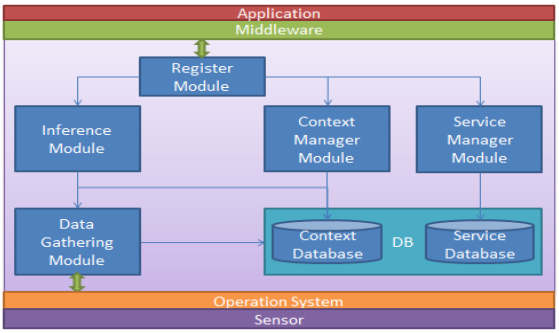


Figure 1. Architecture of the middleware.

The architecture of the middleware is shown in Figure1. This middleware acquires and analyzes context, furthermore, it can provide the upper application with the right services. It is mainly composed of 5 parts: inference module, context manager module, register module, service manager module and data gathering module. Meanwhile, the database is responsible for saving and maintaining the context information and the request ranking information about every application.

A. *Inference module*

Inference Module is responsible for interpreting or aggregating high level context on the basis of the low level context and inferring the situation further, which satisfies the needs of developers better. It receives the request from the register module and informs the data gathering module to collect initial data about the context. Then, according to the context manager module's episode rules, it interprets the low level context which is saved in the database. If there are corresponding situation being matched, then the inference module will informs the register module and the register module will provide callback service.

There are two approaches that are used for reasoning the high level context. One is based on domain knowledge which defined by the experts, such as rule-based inference. Its complexity is very low, but when meeting the complicated situation, it cannot define related rules. The other approach is based on statistic learning, which can analyze and explore the correlation between data and situation by statistic, such as Hidden Markov Model and Bayesian Model. The statistics learning approach is suitable for handling the complicated situation, but it requires high energy consumption and storage space.

In order to offer the accurate and reliable presumable results to the application developers, this paper will adopt the inference model which combines the rule-based inference and the stochastic algorithms of Bayesian Model together.

1) Based on domain knowledge

In the integrated tree structure's expression, the leaf node is data collected by the lower level sensor and the root node is the situation inferred by the model. The user can define a rule to judge the condition by using the expression and the bit arithmetic 'AND', 'OR' and 'NOT'. And the related rule is stored in the context manager module. The inference mechanism can use the situation's ID to acquire the related expression and also traverse all branch of the tree to make sure whether it has been triggered.

2) Based on Bayesian Model

Based on the Bayesian, the inference mechanism can ensure the user's situation through calculating a group probability of the context situation. Equation (1) is the formula of Bayesian.

$$P(C|F1,...,Fn)=\frac{P(F1,...,Fn|C)\times P(C)}{P(F1,...,Fn)} \quad (1)$$

If the probability of the eigenvalues F1,⋯,Fn in the condition of the C is known, then the probability of C in the condition of the eigenvalues F1,⋯,Fn can be known. Assume that eigenvalues are independent. Then, the module of the Bayesian can be obtained.

$$P(C|F1,...,Fn)=\frac{1}{Z}P(C)\prod_{i=1}^{n}P(Fi|C) \quad (2)$$

In the inference module, the user set up a set which contains the situation C and its eigenvalues F1,⋯,Fn. The mechanism classifies the context test sample F1,⋯,Fn, and when the probability of C reaches a maximum, the situation C is triggered.

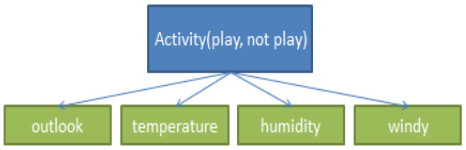
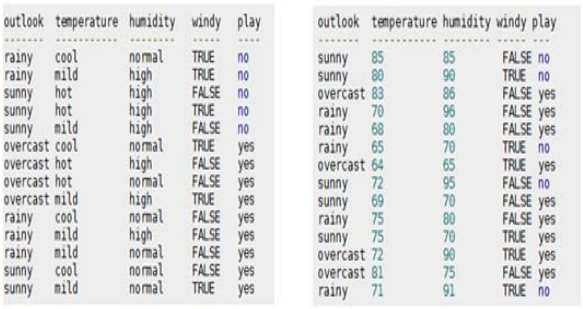


Figure 2. Example of the inference mechanism based on Bayesian Model.

As shown in the Fig.2, based on 4 parameters about weather that is 'outlook', 'temperature', 'humidity' and 'windy', the mechanism infers whether the player can play sport. If this situation is triggered, a training sample can be defined by the vector, which consists of this four characteristics' actual data and the results of the users' situations. And many training samples form the set of the training samples. The mechanism gathers a group of weather's characteristics, then, based on the sample, calculating the probability of the user's choice (e.g. play sport or not play sport). Finally, the triggered situation can be gotten.

There are two methods when inferring which are 'Text' as the feature and 'Numeric' as the feature.



outlook	temperature	humidity	windy	play
rainy	cool	normal	TRUE	no
rainy	mild	high	TRUE	no
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
sunny	mild	high	FALSE	no
overcast	cool	normal	TRUE	yes
overcast	hot	high	FALSE	yes
overcast	hot	normal	FALSE	yes
overcast	mild	high	TRUE	yes
rainy	cool	normal	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	cool	normal	FALSE	yes
sunny	mild	normal	TRUE	yes

outlook	temperature	humidity	windy	play
sunny	85	85	FALSE	no
sunny	80	90	TRUE	no
overcast	83	86	FALSE	yes
rainy	70	96	FALSE	yes
rainy	68	80	FALSE	yes
rainy	65	70	TRUE	no
overcast	64	65	TRUE	yes
sunny	72	95	FALSE	no
sunny	69	70	FALSE	yes
rainy	75	80	FALSE	yes
sunny	75	70	TRUE	yes
overcast	72	90	TRUE	yes
overcast	81	75	FALSE	yes
rainy	71	91	TRUE	no

Figure 3. Training sample

The training sample is shown in the Fig.3.

Assume that the test sample is 'sunny cool normal TRUE', the result is shown in below.

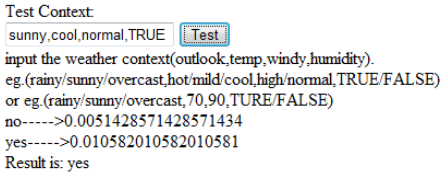


Figure 4. Execution screen when 'Text' as the feature.

Assume that the test sample is 'sunny 66 90 TRUE', the result is shown in below.

Figure 5. Execution screen when 'Text' as the feature.

B. Context Manager Module

The context manager module is responsible for storing and protecting the regular knowledge of the context's inference. The situation's ID identifies every situation's inferred knowledge. This module acquires the regular inferred knowledge which defined by user and the training dataset based on the Bayesian Model which provided by user. Meanwhile, middleware stores some common situation training datasets.

Through traversing the regular knowledge and the test sample's tree structure, the context service can be inferred and saved in the database. Then, through the high-level inference mechanism, the raw data can be acquired for subsequent processing, allowing it to evaluate whether the situation is triggered.

Based on domain knowledge

This module uses the nested structure to model the context tree modeling. Every high-level context situation is identified by Situation ID. The tree structure is defined by some Conditions (e.g. AND, OR and NOT).To every Condition, it is related to the data which is detected by the sensors and through defined Sensor ID, Condition can contact to the sensor. Thus, Condition is the logical comparison between the data from Sensor and Parameter that are some features of the situation and can restrict Condition, and Parameter and Operator (e.g. >, < and =) which are used in there can be defined based on the users' need. Furthermore, in order to provide the precise matching regulation module and reduce the error and collision, the context manager module has an evaluating layer for users, and user can use Specifier, to define the matching module and accuracy. In the evaluating layer, the level of the service request is determined by the sampling period and the number of samples; Mode (e.g. MAX, MIN, MEAN, DIFFERENCE, GRADIENT, etc) can be chosen.

1) Based on Bayesian Model

Based on Bayesian Model, the test sample can be classified by situation.

When the value of the context's test sample is discrete value, through calculating the probability of the discrete value Fi in different situation, the situation's classification can be achieved. When the value of the context's test sample is continuous value, if the sample obeys the normal distributions, conditional probability can be gotten. The formula is shown in below.

$$P(Fi = \mathrm{f\ i} \mid C = c) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\left(\frac{-(\mathrm{f\ i}-m)^2}{2\pi\sigma^2}\right)} \quad (3)$$

In (3), μ represents the mean value of Fi in the situation c; σ^2 represents the variance of Fi in the situation c.

C. Register Module

The register module is the only interface between the middleware and the APP developers. It looks like a listener in the interface. To the upper layer, it provides the customized service to the application, so that the developers don't need to be concerned with the details about the context processing. Besides, through configuring the interfaces' parameters, the user can call other module of middleware and achieve centralized management of the context information. The developers can use middleware as agency and invoke the related API to customize the service.

D. Service Manager Module

The service manager module is designed for saving the energy. It is responsible for handling the service requests of every sensor. After the module receives the updated service requests from the register module, it will use optimizing algorithm to update the value of the corresponding sample and period in the service database. For using the least space to store the same context information of many applications, the service manager module need to adjust precision for sampling and storing, when it receives the same requests. And to the different requests from one sensor, the service manager module needs to calculate a common service for them.

E. Data Gathering Module

The data gathering module is responsible for invoking the sensors of the mobile terminal, then perceiving the

context original data and updating the context database. After the inference module sends the request of gathering the data to this module, it will try to find the data about the service in the database of the service. In addition, this module updates the database of the context based on the number of the samples regularly. Sometimes, the inference module may provide a request from the same sensor repeatedly. At that time, this module will read the value of the context's service level again and adjust the precision of the sampling and storing.

Ⅳ Application – Smart Assistant

The Smart Assistant widget application is developed as the example of the context-aware application to check the availability of the middleware. This application shows how middleware provide the inference to application.

Thus, this application can help the user to analyze context and provide API's suitable interface. Aiming at the different design requirement, developers set the parameter of the context.

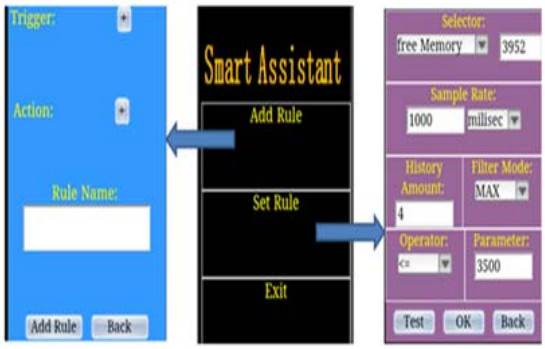


Figure 6. Screen

As shown in the Fig.6, the central is the main screen and the developers can add or set the rule by themselves. And the users can add rules to the register module and then the context manager module acquires the inference knowledge from it. For example, firstly, they choose the type of the trigger, then, define the service of this rule in 'Action'.

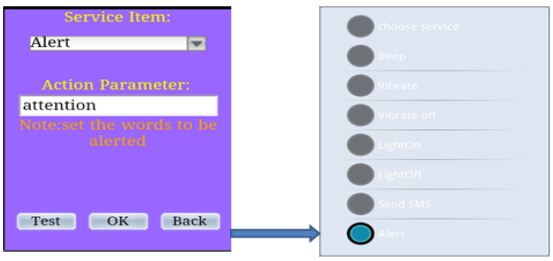


Figure 7. Define the service of the rule

The 'Action' screen is shown in Fig.7. Finally, import the regulation's ID to distinguish others in 'Rule Name'. By this way, a new rule can add in the context manager module. After, the users can define this rule in details. In 'Set Rule', according to their own need, the users set the type of the context, sampling period, 'Mode', 'Parameter' and 'Operator'. Thus, the logical relationship between the 'Parameter' and the sensor data and the service degree can be ensured by this mean.

The final result of the definition of the regulation is shown in Fig.8. And the user should also name the rule in the end.

After these, the user can implement the new rule to check this mechanism, which is shown in the Fig.9. The application listens to this situation and gets the data acquisition and recording of it. At last, the situation is triggered and executes the behavior.

Figure 8. Final result

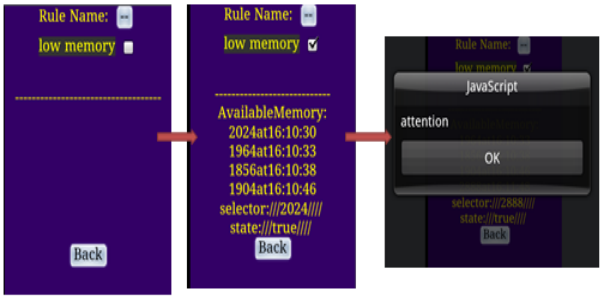


Figure 9. Execution screen

Conclusion

This paper proposes and designs the developing framework of the cross-platform and context-aware middleware application. The proposed middleware can reduce the context's complexity, simplify the development process of the context application and solve the compatibility problem in different applications' platform. In addition, this middleware provides two inference mechanisms that satisfy the developers' needs of setting the inferences and provides the situations' classification method when the logical inference is incomplete.

Reference

[1] Mark Weiser, "Some Computer Science Issues in Ubiquitous Computing", Communications of the ACM, pp.75-84, 2003.

[2] Mao Yanhua, "The research of the software platform of smart space and its resource management ", Beijing:Tsinghua University computer science and technology, 2004.

[3] "MIT project oxygen",http://oxygen.lcs.mit.edu/Overview.html

[4] "IBM research", http://www.research.ibm.com/natural/dreamspace

[5] H. Chen, F. Perich, T. Finin, and A. Joshi. “SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications”. In Int. Conf. on Mobile and Ubiquitous Systems: Networking and Services, Aug.2004.

[6] Joonas Kukkonen and Eemil Lagerperx, “BeTeiGeuse:a platform for gathering and processing situational data”, IEEE Pervasive, pp49-56, 2009