A New Context Script Language for Developing Context-Aware Application Systems in Ubiquitous Computing*

Jae-Woo Chang and Yong-Ki Kim

Dept. of Computer Engineering
Chonbuk National University, Chonju, Chonbuk 561-756, South Korea
jwchang@chonbuk.ac.kr, ykkim@dblab.chonbuk.ac.kr

Abstract. In order to develop a variety of context-aware application systems, we require a context script language to describe both various decisions on context-awareness and appropriate procedures according to the decision. In this paper, we propose a new context script language which can represent a variety of contexts as a standard syntax. The proposed context script language is a general purpose one to provide users with functions to define a given context in a clear and precise manner. To show the usefulness of the proposed context script language, we develop a context-aware application system using it, which can provide users with a music playing service in ubiquitous computing environment.

1 Introduction

Ubiquitous computing is embedded in the users' physical environments and integrates seamlessly with their everyday tasks [1, 2]. An effective software infrastructure for running ubiquitous computing applications must be capable of finding, adapting, and delivering the appropriate applications to the user's computing environment based on the user's context. Thus, context-aware application systems determine which user tasks are most relevant to a user in a particular context, and they may be determined based on history, preferences, or other knowledge of the user's behavior, as well as the environmental conditions. In order to develop a variety of context-aware application systems in an effective manner, we require a context script language to describe both various decisions on context-awareness and appropriate procedures according to the decision.

In this paper, we propose a new context script language which can represent a variety of contexts as a standard syntax. The proposed context script language is a general purpose one to provide users with functions to define a given context in a clear and precise manner. To show the usefulness of the proposed script language, we develop a context-aware application system using it, which can provide users with a music playing service in ubiquitous computing environment. The remainder of this paper is

* This work is financially supported by the Ministry of Education and Human Resources Development(MOE), the Ministry of Commerce, Industry and Energy(MOCIE) and the Ministry of Labor(MOLAB) though the fostering project of the Lab of Excellency.

X. Zhou et al. (Eds.): EUC Workshops 2006, LNCS 4097, pp. 43 – 52, 2006.

organized as follows. The next section discusses related work. In section 3, we describe the design of our context script language. In section 4, we present the development of our context-aware application system. Finally, we draw our conclusions in section 5.

2 Related Work

In this section, we introduce some related work on context description languages. First, Arizona State Univ. [3] presented a context definition language called CA-IDL (Context-enabled Interface Definition Language). That is, context tuples for defining context can be represented as Context tuple : < a_1, ... , a_n, t_m>. Here, n means the number of unique context data, a_i means the value of the i-th context data, and t_m means the time for tuple creation. For example, if the context tuple consists of location, direction, and velocity, it can be represented as <(x,y), north, m, t> when its context object is moving toward north. Secondly, INRIA in France [4] proposed a general infrastructure based on contextual objects to design adaptive distributed information systems in order to keep the level of the delivered service despite environmental variations. The contextual objects (COs) are mainly motivated by the inadequacy of current paradigms for context-aware systems. The use of COs does not complicate a lot of development of an application, which may be developed as a collection of COs. The COs are defined as CO(id) : <Variant V_x Attribute A_i: value, Attribute A_j: value, ...>. For example, two contextual Web documents can be defined as <V_1; Location:L_1, Language:French, Browser:with frame> and <V_2; Location:L_2, Language:English, Browser:without frame>. The Web documents can be browsed by selecting its appropriate variant according to users' location, users' language, and browser type.

3 Context Script Language for Context-awareness

In this section, we will propose a new context script language to represent contexts precisely and describe a grammar specification for it.

3.1 Components of Context Script Language

A context object is any logical concept that can be used to characterize the situation of any entity [5]. 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 applications themselves. In order to develop a variety of application software for context-awareness in an effective manner, a context script language is required to describe both various decisions on context-awareness and appropriate procedures according to the decision. Even though the CA-IDL was proposed as a context script language, it is not a general purpose one because of being dependant on a specific system. Thus, we propose a new context script language which can represent a variety of contexts as a standard syntax. The context script language is a general purpose one to provide users with functions to describe a given contexts in a clear and precise manner. Our context script language has such components as context object definition, context creation & destroy, context instance insertion & deletion, and context instance activation & deactivation.

3.1.1 Context Object Definition

A context object is a logical concept that can be used to represent any entity or device around application software for context-awareness. Some attribute values for a context can be updated by hardware devices, which can be used in a conditional clause to be mentioned in the next. The definition clause of the context objects is shown as follows. Here object_name and element_name never begin with a numeric or special letter, and they have less than 256 in length. In addition, data_type consists of two types; string data type (i.e., string) and numeric data type (i.e., int, float, long, double, time, date).

```
ContextObject object_name (
    data_type element_name1,
    data_type element_name2,
    data_type element_name3
              :              )
```

3.1.2 Context Creation and Destroy

A context creation makes a rule to execute an action when a context satisfies a given condition. It consists of a conditional expression and an action being executed. The context creation defines the template of context table and the values of context table are actually inserted in case of inserting a context instance. The clause of the context creation is shown as follows, where context_name being less than 256 in length cannot begins with a numeric or special letter. Here a condition clause is used to represent a given condition while action and parameter clauses are used to indicate both a function and its parameters being called when satisfying the condition.

```
ContextCreate context_name
Condition    (condition_expression) [during time_value]
Action       function_name
             [before/when/after time_value] or
             [from time_value to time_value]
Param        (param1, param2, ... )
```

First, the conditional expression (i.e., condition_expression) can be composed of N conditional clauses and N-1 logical operators (i.e., AND, OR, NOT). The conditional clause has two operands and one relative operator (i.e., =, >, <, <=, >=). The during time_value clause being optional can be used to represent a condition related with time. The time_value expresses a time interval with s(second), m(minute), and h(hour). For example, if you need a time interval for 1 hour and 12 min and 30 sec, it can be expressed as 1h12m30s. Thus, the condition that a person P locates in A or B place for 1 minute and 30 seconds can be expressed as (((P_person.location = A_place) during 10s) or ((P_person.location = B_place) during 1m30s)). Secondly, the function_name means the name of a function to be executed when a given condition is satisfied. If there is no need to call a function, NULL can be used as the function_name. The optional clauses of [while/when/after time-value] and [from time_value to time_value] can be used to determine when and how long an action should be executed. For the while/when/after time-value, an action should be

executed once before, exactly when, and after a given time, respectively. For the from time_value to time_value, an action should be executed continuously during the time (from time_value to time_value). Figure 1 shows time relationship in the action clause.

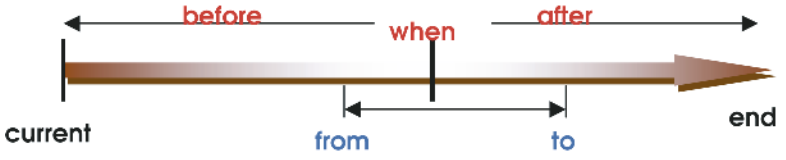

Fig. 1. Time relationship in the action clause

Finally, Param (param1, param2, ..) means a list of parameter names which are used in a function being executed by Action. If there is no parameter, NULL can be used for parameter names. The created context can be destroyed by the clause of ContextDestroy as follows. When a context is destroyed, all the instances of the context are deleted and a status for checking a condition to activate the context is disabled.

ContextDestroy context_name

3.1.3 Context Instance Insertion and Deletion

Once a context is created, it is necessary to add context instances into the context. The clause of context instance insertion is shown as follows, where ContextInsert, Condition, and Param are used. Here the context_name means the name of a context which the instance will be inserted into, while the instance_name means the name of a context instance to be inserted. The condition_value, which has a constant or constant literal, is used to instantiate a variable in the condition_expression for creating a context. Similarly, the parma_value has a real value for a parameter of a function being called by Action in the context creation clause.

ContextInsert instance_name Into context_name
Condition (condition_value1, condition_value2, ...)
Param (param_value1, param_value1, ...)

The context instance inserted can be deleted by the clause of ContextDelete as follows. Here the context_name means the name of a context which the instance will be deleted from, while the instance_name means the name of a context instance to be deleted.

ContextDelete instance_name In context_name

3.1.4 Context Instance Activation and Deactivation

Once a context is created and its context instances are inserted, the activation of a context instance makes it possible to check a condition and execute an appropriate action when satisfying the condition. On the contrary, the deactivation of a context

instance makes it impossible to execute an action by checking its corresponding condition. The clause of context instance activation and deactivation is shown as follows.

ContextActiv instance_name In context_name

ContextDeAct instance_name In context_name

Here the context_name means the name of a context whose instance will be activated or deactivated, while the instance_name means the name of a context instance to be activated or deactivated. In addition, the asterisk symbol (*) can be used when all the instances in a context are activated or deactivated.

3.1.5 Example of Context Script Language

As an example of a user's context, his (or her) popular music is used where each user is moving from room to room. When someone is staying in a room for 2 seconds, he (or she) can hear his (or her) popular music playing in the room after one second. This example can be described by using our context scrip language, where the bold name means reserved words as shown in Figure 2.

```
ContextObject person (
      string name,
      string location );
ContextCreate music
Condition ((person.name = someone_name) AND
            (person.location = someone_location) During 2s)
Action PlayMusic When 1s
Param (music_name)
```

Fig. 2. Example of a user's context as his (or her) popular music

In the example, we suppose that when a person named 'Chul-Su Kim' enters into a room 7401, a music named 'Arirang' is playing while when he is in a room 7429, a music named 'Doragi' is playing. The context instance insertion for this example can be expressed as Figure 3, where someone_name, someone_location, and music_name in Figure 2 are instantiated as 'Chul-Su Kim', 7401, and 'Arirang', respectively.

```
ContextInsert kim Into music
Condition ('Chul-Su Kim', '7401 ')
Param (' Arirang ');
ContextInsert kim Into music
Condition (' Chul-Su Kim ', ' 7429 ')
Param (' Doragi ');
```

Fig. 3. Example of context instance insertion

We have to check a condition to activate all the context instances of music, by using the context activation clause as follows.

ContextActiv * In music

3.2 Grammar Specification of the Context Script Language

Reserved words used in our context scrip language are composed of script commands, subsidiary words, time-related words, relative operators, logical operators, constants, comment symbols, and system-defined data types. Table 1 shows the reserved word of our context scrip language. To describe a grammar specification for script commands in the context scrip language, we make use of Backus normal form (BNF). White spaces are used for the readability of the grammar specification using BNF.

Table 1. Reserved words

description	reserved words
script commands	ContextObject, ContextCreate, ContextDestroy, ContextDelete, ContextInsert, ContextActive, ContextDeact
subsidiary words for building script commands	Condition, Action, Param, in, into
time-related words for building script commands	during, when, while, after, period
relative operators	=, <>, <,>, <=, =>
logical operators	and, or , not, AND, OR, NOT
constants	NULL, TRUE, FALSE, null, true, false
system-defined data types	string, int, float, long, double, time, date

3.2.1 ContextObject

An element for a context object consists of data_type and variable. Thus, a statement to define a context object can be defined as follows.

command context_name [(] element ([,]element)* [)]
element : data_type variable

3.2.2 ContextCreate and ContextDestroy

A conditional expression (i.e., condition_expression) can define not only a simple conditional clause, but also a set of conditional clauses with logical operators (i.e., AND, OR, NOT). In addition, a conditional clause belongs to one of the following four types, such as the comparison between the value of a context instance and a variable, the comparison between the value of one context instance and that of another context instance, the comparison between two variables, and the comparison between a variable and the value of a context instance.

condition_expression : [(] condtion[lo_op condition]* [)]
condition : context_name[.]context_obj_name re_op var |
context_name[.]context_obj_name re_op
context_name[.]context_obj_name |
var re_op var |
var re_op context_name[.]context_obj_name

Thus, a statement to create a context can be defined as follows.

command context_name
command_ad condition_expression (command_time time_value){0,1}
command_ad func_name (command_time timevalue){0,1}
command_ad [(] func_name ([,]func_name])* [)]

The context created can be destroyed by using ContextDestroy as follows.

command context_name

3.2.3 ContextInsert and ContextDelete

A clause for inserting a context instance is defined as follows. Here the clause is used to insert real values into the variables in the Condition and the Param clauses.

command context_ins_name command_ad context_name
context_ad [(] value_const ([,]value_const)* [)]
context_ad [(] value_const ([,]value_const)* [)]

A clause for deleting a context instance is defined as follows.

command context_ins_name command_ad context_name

3.2.4 ContextActive and ContextDeact

A clause for activating the context instance inserted is defined as follows. Here the clause for Period can be optional.

command context_ins_name command_ad context_name (command_time time value){0,1}

A clause for deactivating the context instance inserted is defined as follows.

command context_ins_name command_ad context_name

4 Development of a Context-Aware Application System

In order to show the efficiency of our context script language, we develop a context-aware application system which provides users with a music playing service in ubiquitous computing environment. Figure 4 shows an overall architecture for supporting our music playing service.

Here, when a user belonging to a moving node approaches to a fixed node, the fixed node starts playing the user's music with his (her) preference according to his location. In general, each user has a list of his (her) music with his preference and even a user can have a different list of his (her) popular music depending on time, i.e., morning time, noon time, and night time. For example, when a user K hearing his music with his preference in the place A moves to the place B, the fixed node 1 stops playing his music in the place A while the fixed node 2 starts playing his music in the place B.

We develop our context-aware application system providing a music playing service by using affix 2.0.2 as a Bluetooth device driver protocol stack [6] and by using GCC 2.95.4 an a compiler, under Redhat Linux 7.3 (kernel version 2.4.20) with 866

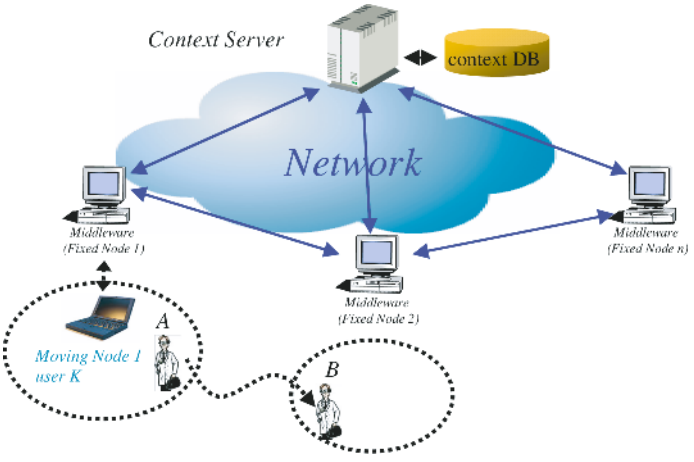

Fig. 4. Overall architecture for supporting context-aware application services

MHz Pentium-III CPU and 64 MB main memory. In addition, the Bluetooth device follows the specification of Version1.1/Class1 and makes a connection to PCs using USB interfaces [7]. To determine whether or not the context-aware application system implemented works well, we test it by adopting a scenario used in Cricket [8], one of the MIT Oxygen project. For this, we test the execution of our context-aware application system in the following three cases; the first case when a user covered by a moving node approaches to a fixed node or move apart from it, the second case when two different users approaches to a fixed node, and final case when a user approaches to a fixed node at different times. Among them, because the first case is the most general one, we will explain it in more detail. For our testing environment, we locate two fixed nodes in the database laboratory (DB Lab) and the media communication laboratory (Media Lab) of Chonbuk National University, respectively, where their middleware can detect a moving node by using Bluetooth wireless communication. There is a corridor between DB Lab and Media Lab and its distance is about 60 meter. We test the execution of our middleware in case when a user having a moving node moves from DB Lab to Media Lab or in a reverse direction. Figure 5 shows a testing case when a user having a moving node approaches to a fixed node. First, the fixed node receives a user name from the moving node as the moving node is approaching to it (①). Secondly, the fixed node determines whether the information of the user has already been stored into a server or not. If the information has been there, the context server searches the music file belonging to the user in a current time and downloads the music file from the database (②). Finally, the fixed node starts playing the downloaded music file using a MP3 player (③). Similarly, we will explain a testing case when a user having a moving node moves apart from a fixed node. First, when the fixed node detects that a user is too far from the fixed node to communicate with it, the fixed node stops the process to play music and removes the music playing process. Conclusively, when a user approaches to a fixed node, the fixed node starts playing the user's music while when a user moves apart from the fixed node, the fixed node stops playing the music.

To analyze the performance of our context-aware application system, we measure an average time by adopting a boundary detection of beacons used in Cricket. Table 2 shows the average time to aware contexts. First, as a moving node is approaching to a middleware, it takes 1.34 second to make a connection between them. It means the time for the fixed node to detect the presence of a moving node when a moving node enters into its communication boundary. The time mainly depends on the specification of Bluetooth wireless communication. Secondly, it takes 0.5 second for the fixed node to start music playing service after making the connection between them. It means the time for the fixed node to search the profile of the corresponding user and to call the module to play music. The searching time for a user's profile is dependant both on the packet transfer time of TCP/IP and on the DBMS performance of the context server. The calling time for the music playing module means the one for an operating system (OS) to load it, which is affected by the available memory of the OS kernel and the speed of a hard disk. Finally, as a moving node is going apart from a fixed node, it takes 1.45 second to make a disconnection between them. It means the time for the fixed node to detect the absence of the moving node. The time is relatively long because the kernel tries to communicate with the moving node even though the moving node is beyond the communication boundary of the fixed node. Therefore, when the kernel is disconnected with the moving object, it is very reasonable for the fixed node to set a time limit to two seconds. If it takes long time for a

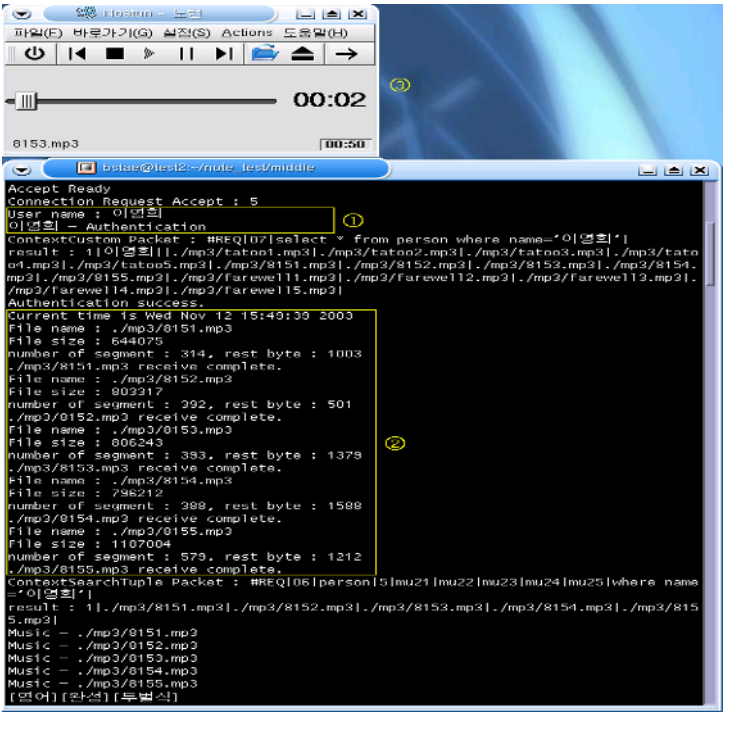

Fig. 5. Testing case when a user approaches to a fixed node

Table 2. Average time for connection and disconnection

activities	Time(sec)
Average time for a middleware to make a connection to a moving node	1.34
Average time for a middleware to start a music playing service	0.50
Average time for a middleware to make a disconnection to a moving node	1.45

fixed node to establish a connection to a moving node and to detect a context from it, a user may consider the situation as a fault. Because the detection time for a context is less than two seconds, our context-aware application system is reasonable for the music playing service in ubiquitous computing environment.

5 Conclusions and Future Work

In this paper, we proposed a new context script language which can represent a variety of contexts as a standard syntax. The proposed context script language is a general one to provide users with functions to define a given context in a clear and precise manner. To show the usefulness of the proposed script language, we developed our context-aware application system which can provide users with a music playing service in ubiquitous computing environment. We tested our context-aware application system by adopting a scenario used in Cricket, one of the MIT Oxygen projects. It was shown that it took about 1.5 seconds to make a connection (or disconnection) between a fixed node and a moving node, thus being considered reasonable for our music playing service. As future work, it is required to study on a rule description language which can be used to express the inference of a new context from the existing contexts.

References

1. G. Banavar, A. Bernstein, "Issues and challenges in ubiquitous computing: Software infrastructure and design challenges for ubiquitous computing applications", Communication of ACM, Vol 45(12), pp. 92-96, 2002.
2. M. Weiser, "Some Computer Science Issues in Ubiquitous Computing", Communication of the ACM, Vol 36(7), pp. 75-84, 1993.
3. S. S. Yau and F. Karim, "Context-sensitive Middleware for Real-time Software in Ubiquitous Computing Environments", Proc. of 4th IEEE Symposium on Object-oriented Real-time Distributed Computing, pp.163-170, 2001.
4. P. Couderc, A. M. Kermarrec, "Improving Level of Service for Mobile Users Using Context-Awareness", Proc. of 18th IEEE Symposium on Reliable Distributed Systems, pp. 24-33, 1999.
5. A. K. Dey, "Understanding and Using Context", Personal and Ubiquitous Computing Journal, Vol. 5, No. 1, pp. 4-7, 2001.
6. Affix: Bluetooth Protocol Stack for Linux, http://affix.sourceforge.net.
7. Bluetooth Version 1.1 Profile, http://www.bluetooth.com.
8. N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The Cricket Locaion Support System", 6th ACM/IEEE Int'l Conf. on Mobile Computing and Networking(MOBICOM), pp. 32-43, 2000.