A Service Control Method Using Semantic Constraint on Context Interpretation

Jun Maeda, Masaaki Takase, and Mitsuaki Kakemizu

***Abstract*— When constructing a personalized context-aware service system, much work is required to develop the service control programs that provide services in various personalized contexts. Then, a common control system that interprets the context and dynamically generates various service execution methods is necessary. A platform that generates service execution methods by combining context interpretation information is proposed. The semantic constraint is introduced as a part of the service definition. It generates appropriate service execution methods by specifying how to use the context and it also improves processing performance.**

***Index Terms*— context-aware, context interpretation, service control, service definition**

I. Introduction

In the age of ubiquitous and personal services, context-aware services [1]-[2] that provide suitable services for individual user's situations (context) are expected. Context-aware service control platforms [3]-[6] have been developed as basic facilities for such as acquisition, sharing and use of the context. To provide a context-aware service, a service execution method that achieves the user's objective under the precondition of context is necessary. The service execution methods are described in the application programs on the platform, however because the service execution methods differ according to the context even if the user's objective of the service is the same, it is necessary to consider various contexts in the application programs, and much work is required for the program development. It therefore becomes necessary for the service execution methods to be generated dynamically. This research aims to develop a function to reactively generate personalized service execution methods based on the context as a higher layer function of the context-aware service control platforms.

II. Problem

The function to generate the service execution method from the context and the user's objective is termed "context interpretation" in this paper. The context interpretation is performed by deriving related information about the service control necessary to achieve the objective from the context. So far, context interpretation processing is usually realized by the following method 1.

Method 1 (individual application processing)

In this method, each application program performs context interpretation. The typical processing sequence is as follows: (1) the application registers trigger information in the platform; (2) the platform calls the application when the context matches the trigger information; (3) the application generates the service execution method by interpreting the context; and (4) service control action is performed according to the service execution method. Existing platforms such as the CASC (Context-aware service control) system [3]-[4] are designed with such a processing style, however this method requires a variety of context interpretation programs in the application program because of various personalized contexts. As a result, the problem of the expanding application program development emerges. Fig. 1 shows an example of the application program interpreting context using method 1. While it is the case that the context interpretation is performed in the platform, if information on the service execution method for an individual service is provided by the application, this is classified as method 1.

```
if Context1 derives information A by relation R4
{
   if Information A derives service objective D by relation R2
      A service execution method is found.
   if Information A derives service objective E by relation R3
      A service execution method is found.
}
if Context1 derives information B by relation R1
{
   if Information B derives service objective D by relation R2
      A service execution method is found.
   if Information B derives service objective E by relation R3
      A service execution method is found.
}
if Context1 derives information C by relation R11
   if Information C derives service objective D by relation R2
      A service execution method is found.
if Context1 derives information L by relation R11
   if Information L derives service objective M by relation R5
      A service execution method is found.
```

Fig. 1. Context interpretation program by Method 1

Jun Maeda is with Fujitsu Limited, Kawasaki, 211-8588 Japan (e-mail: maeda.jun@jp.fujitsu.com).

Masaaki Takase is with Fujitsu Limited, Kawasaki, 211-8588 Japan (e-mail: masaaki.takase@jp.fujitsu.com).

Mitsuaki Kakemizu is with Fujitsu Limited, Kawasaki, 211-8588 Japan (e-mail: kake@jp.fujitsu.com).

It is essential that the platform generates service execution method for personalized context-aware services. We then adopted the following method 2.

Method 2 (common processing)

This method performs context interpretation by common processing. The processing sequence is as follows: (1) the application registers trigger information in the platform; (2) the platform generates the service execution method by interpreting the context; and (3) the platform calls applications that control service action according to the service execution method.

A platform that constructs a user profile to be used by applications was proposed [5]. In this system, the control of individual services is performed by the application program (method 1), but the platform performs a part of context interpretation that does not depend on individual services. If this idea is expanded for the entire context interpretation, the service execution method can be constructed from the context. This is performed by looking for the "story" that reaches the service objective by combining context interpretation information (various information derived from the context) with the context. Fig. 2 shows a sample of context interpretation.

A mechanism that enables flexible and efficient context interpretation is required. It is also necessary to make service development simple.

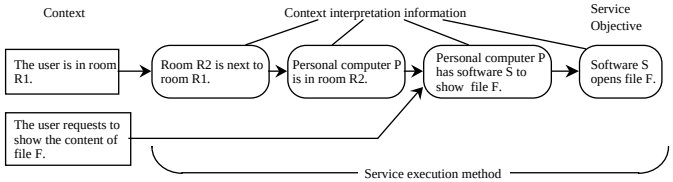

Fig. 2. A sample of context interpretation

III. PROPOSED CONTEXT-AWARE SERVICE CONTROL METHOD

We designed a context interpretation mechanism as follows.

1) It is necessary that the mechanism be able to generate diverse service execution methods to provide services by flexible means. Essential information that is in the context or the context interpretation information is then used to combine the context interpretation information. For instance, essential information items in Fig. 2 are user's location R1, room R2, personal computer P and so on. This method deals with every item of context interpretation information that has a possibility of being combined, then it can generate varied service execution methods.

2) It is not that the related context interpretation information is simply combined with the context, but the context interpretation information suitable for the service objective should be combined. The semantic constraint then selects the context interpretation information suitable for the service objective. In Fig. 2, for instance, context interpretation information "There is an air-conditioner A in room R2" can be combined with context interpretation information "The room next to room R1 is room R2" through "room R2" as essential information. However, if the semantic constraint for the objective "file opening" specifies "derive information about IT equipment," "air-conditioner A" does not satisfy the semantic constraint and this context interpretation information is not selected. This method generates semantically appropriate service execution methods, and reduces context interpretation processing by eliminating the search process for meaningless service execution methods.

Fig. 3 shows the entire mechanism. The following describes the explanations of each part, details of context interpretation processing and discussions about the proposed method.

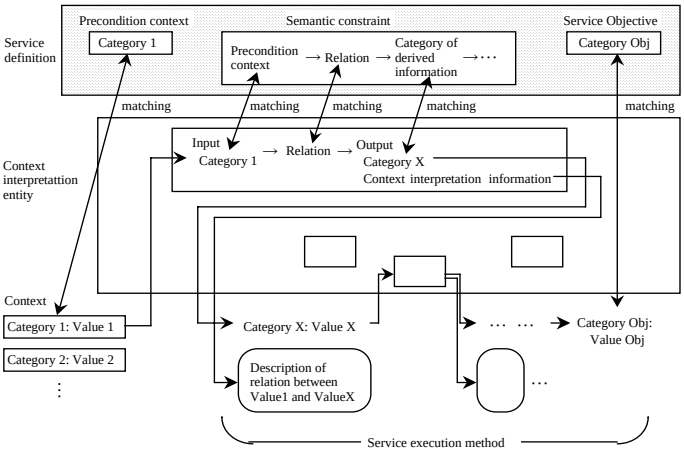

Fig. 3. Service control mechanism using semantic constraint

A. Context

The context is information that shows various situations of the real world. The context includes information supplied by sensors or information providers, etc. Information that a user enters from a personal terminal (parameters for specific service, and personal information such as location of user, age, hobbies, etc.) is also included in the context. The context is represented by category (type of information) and value. The following are examples of context: "User's location: conference room," "Time: ten o'clock," "Document name specified by user: document_1" and "User's address: Tokyo".

B. Context interpretation information

This information gives the interpretation of the context and is provided from the context interpretation entities. When the context interpretation entity receives an input value that belongs to a specific category, it then outputs a value that belongs to a specific category and context interpretation information according to semantic relations between the input and output defined for each entity. For instance, a context interpretation entity receives the input value "User's location: R1," then outputs the value "Room: R2," and context interpretation information "Room R2 exists on the same floor

as R1" according to a semantic relation "Existing on the same floor."

C. Service definition

The service definition consists of the precondition context, the service objective, and the semantic constraint. This describes "What action is performed (service objective) by using which information (precondition context) and how such information is used (semantic constraint)." Service objective describes the category that shows the service action such as "Document edit" and "Air-conditioning control." Precondition context describes the category of the information used in the service such as "User's location" and "User specified document." Semantic constraint is provided for each precondition context. This defines the scope of the semantic relation between the precondition context and the context interpretation information that it is combined with, and the scope of category which that derived value belongs to in the context interpretation processing. Similarly, it defines the constraint of context interpretation information combined with derived information. Semantic constraint of the precondition context not used to derive any information is indicated as "Direct use".

Examples of semantic constraint are shown below.

1) Precondition context: User's location -> Relation: Existing in same place -> Derived information: Equipment.
2) Precondition context: User's location -> Relation: Existing in same place -> Derived information: Personal computer.
3) Precondition context: User's location -> Relation: Existing on the same floor -> Derived information: Place -> Relation: Existing in same place -> Derived information: Equipment.
4) Precondition context: User's location -> Relation: Attribute -> Derived information: Temperature.
5) Precondition context: User specified document name -> Relation: Direct use.

A hierarchical structure is used for representing relations in different abstraction levels, from general relations to specific relations. Similarly, the categories of derived information use a hierarchical structure.

D. Context interpretation processing

In context interpretation processing, the present context is first compared with the precondition context that is described in the service definitions registered for each user, and if the precondition context of a service definition matches the present context, the definition is activated. The context interpretation information is then combined based on the activated service definition using graph search processing.

Combining context interpretation information is performed as follows. First the context interpretation entity that receives a value belonging to a category that matches the precondition context and is consistent with semantic constraint is invoked. Context interpretation information provided from the context interpretation entity is combined with the context. More context interpretation information is combined by repeating this process (next context interpretation entity receives a value that matches the output value of the previous entity). If output of a context interpretation entity matches the category described as a service objective, a set of combined context interpretation information becomes a service execution method. This describes, for instance, "Run software S on personal computer P which is in the room next to user's location to achieve the service objective that file F is shown". Generated service execution method is reliable in terms of semantic appropriateness for the service objective by satisfying semantic constraint.

Context interpretation also can be personalized according to the situations like user's location or behavior. The following is one method.

1) Context interpretation entities are classified into groups, such as general, specific field, district, office and home.
2) Groups suitable for the user's situation are selected and used for context interpretation.

This provides services by context interpretation suitable for a user's situation and efficient processing by eliminating unnecessary context interpretation information. For instance, users at different locations can use services peculiar to each place by selecting the context interpretation entity group for each district, and the processing becomes efficient by not using context interpretation information for other districts.

E. Effect of proposed method: efficient context interpretation

The amount of processing becomes enormous to find every possible service execution method when the context amount and the interpretation information increase. Fig. 4 shows the context interpretation processing by the platform corresponding to the application program in Fig. 1. As shown in Fig. 4, the context interpretation result includes a lot of meaningless service execution methods, because there are many context interpretations which are not relevant for the service objective.

This unnecessary context interpretation processing is reduced by semantic constraint. That is, because context interpretation entities that are not consistent with semantic constraint are not invoked, meaningless combinations of context interpretation information provided by those entities are omitted. In Fig. 4, "X" shows the points where context interpretation information is not combined.

F. Effect of proposed method: simple service development

The service definition method using semantic constraint enables the development of services more simply and easily than method 1 because it is not necessary to describe a detailed context interpretation procedure. Semantic constraint uses hierarchical representations of relations and categories of derived information as shown in Fig. 4. This enables service definitions to be described with the necessary abstraction level from the user's viewpoint. Context interpretation generates various service execution methods suitable for the context

using the graph search algorithm within the scope of a specified abstraction level, thus the amount of description necessary to define the service is significantly reduced compared with the description for the application program in method 1. The service definition described in Fig. 4 is equivalent to the application program in Fig. 1.

On the other hand, if detailed relations and categories are specified in the semantic constraint, it is also possible that the service definition describes a precise procedure similar to method 1.

The service definition composed from service objective, precondition context and semantic constraint seems to be close to the ideas considered when the designers develop new services.

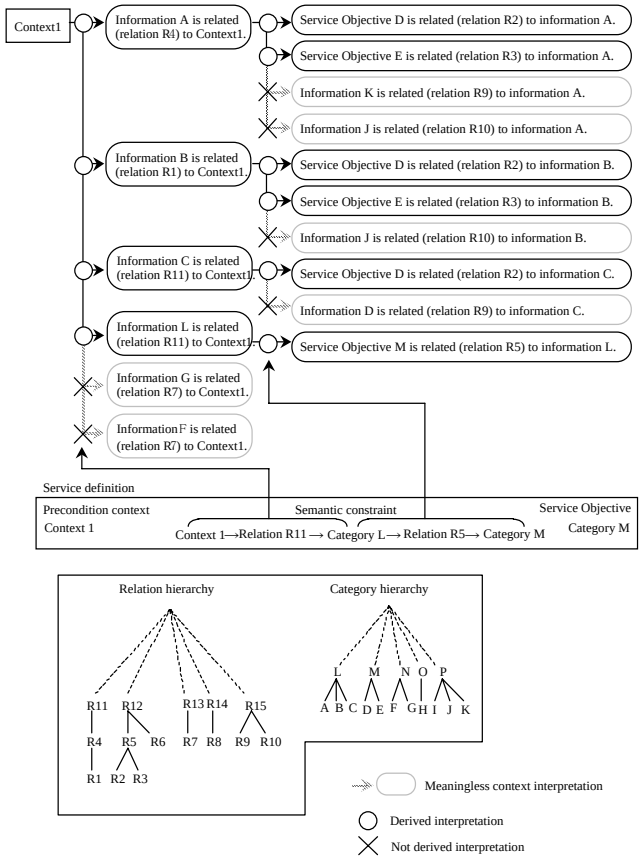

Fig. 4. Context interpretation by Method 2 with semantic constraint

G. *Service conflict*

When services are executed, the requests of users may cause a conflict. A pattern of the conflict is the case that the effect of service Sa, which is based on the context of user A, is removed by other services (for instance, service Sb, which is based on the context of user B) that are in effect at the same time. This brings inconvenience to user A. Another pattern is the case where the execution of service Sc, which is based on the context of user C, alone brings some inconvenience to user A.

Fig. 5 shows an idea to resolve the conflicts in the framework of the service execution generation system. Other people related to a service execution method (for instance, a person in the place where the service is executed or the owner of something used for the service) and their personal requests are derived from each entity that provides those items of information. Conflict resolution entities then decide the appropriate action (resource control) by considering other people's requests. The conflict can be resolved by combining those entities with the service execution method in context interpretation processing. Since the processes of conflict resolution depend on every detail of conflict, it is required to provide corresponding conflict resolution entities.

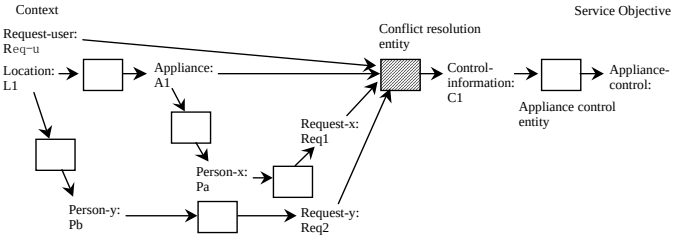

Fig. 5. Conflict resolution

IV. Conclusion

A context-aware service provides customized services for each user according to the dynamically changing personalized context. This paper proposes a basic mechanism for a context-aware service control platform. The fundamental idea is to generate an appropriate service execution method using the service definition represented by the service objective, precondition context and semantic constraint, together with the context interpretation processing that derives the required action from the precondition context based on semantic constraint. This method can efficiently generate a service execution method suitable for the context. We hope to apply this method to the actual environment and to verify it.

This research is the result of contract research on "Ubiquitous network control and management technologies" for the Ministry of Internal Affairs and Communications.

References

[1] B. Schilit, N. Adams and R. Want, "Context-aware computing applications," IEEE workshop on mobile computing systems and applications, 1994

[2] G. Chen and D. Kotz, "A survey of context-aware mobile computing research," Dartmouth Computer Science Technical Report TR2000-381, 2000

[3] Y. Igarashi, M. Takase, M. Kakemizu, and M. Wakamoto, "Location information control architecture for ubiquitous services platform," Technical Report of IEICE, NS2003-59, 2003

[4] M. Takase, Y. Igarashi, H. Takeyoshi, and M. Kakemizu, "Situation-dependent service activation and resource handling method for context-aware application and services," Technical Report of IEICE, NS2003-243, 2004

[5] D. Morikawa, M. Honjo, N. Kotsuka, A. Yamaguchi and M. Ohashi, "Profile aggregation and dissemination; a framework for personalized service provisioning," Ubicomp 2004 Workshop, 2004.

[6] M. Minami, K. Sugita, H. Morikawa, and T. Aoyama, "A design of internet application platform for ubiquitous computing environment," IEICE Trans. Communications (Japanese Edition), Vol. J85-B No.12, 2002.