2005－UBI－8（48）
2005／6／10

Document Based Context Aware Computing

Kazuki Yokoi* **Tsubasa Unemura*** **Shigeo Fujiwara†**
Nobuhiko Nishio‡

ABSTRACT
In recent years there has been a lot of research in context aware computing. Most research generates the context through the combination and abstraction of information obtained from sensors. However higher abstraction level contexts cannot be generated if only sensors are used.

Therefore, we have developed an approach that treats context history and the current context at a high level of abstraction by not only using sensor information but also by allowing the user to actively take part in context recognition, by transmitting and thereby specifying the context.

In this paper, we explain how to semanticaly describe the user's tasks such as reading and writing mail, business manuals as well as knowledge of the users past actions. In addition, we propose a method to recognize context using the arrangement of user tasks.

1. Introduction

With the development of wireless technology and miniaturization of high-powered computers, connecting all objects to the network and embedding computation function ality inside every day environment is becoming possible. Such a computing environment is refered to as ubiquitous computing, and it has been researched to support intelligent and cooperative user task since the beginning of the 1990's [1]. Context aware computing is one area of research concerning such ubiquitous computing environment. The purpose of context aware computing is to reduce the complexity of the process by which services are presented and provided to users.

There are a lot of approaches [2, 3] concerning sensor based context aware computing that recognizes the context by using sensors. Because many of them define context as being obtained momentarily from the sensors, it is difficult for them to recognize context history. Further, because information obtained from the sensors is low level, even if information obtained from two or more sensors is combined, it is difficult to recognize high abstraction level contexts such as a meeting.

We have feel that this kind of context recognition has limitations. Therefore, we are conducting research based on the idea that the user is actively involved in the recognition of context and explicitly passes context to the system.Because the user is actively involved in the recognition of context, the system can recognize a higher level of information when compared to using sensors alone. However, the user's role in context recognition presents the risk of transfering the workload to the user. Our research solves this problem by recognizing context based on the user's regular activity.

How to advance user tasks is often decided by mail, business manuals as well as past knowledge of user activity in daily life. Users look at such an arrangement of tasks, acknowledge the tasks assigned to them, performs them, and eventually completes them as they go about their day. In such a daily life, if the user can: describe machine readable arrangements of user tasks, pass them to the system, and notify the system of completed tasks, the system can recognize: user tasks which a user can face, what kinds of tasks the user did up until now, and what kinds of tasks a user will do from now on. We're contemplating the possibility of using high-level abstraction of context in systems because the user themself describes the arrangement of user tasks. Moreover, we think that it will become possible to use context history because, based on the past, the system knows the arrangements of user tasks to use in the current and future. Therefore our research examines techniques of describing the arrangement of user tasks as machine readable, and a framework that binds a context to services by being passed arrangements of user tasks and notification of task completion from user.

Moreover our research is aiming at application to groupware to be possible to describe the arrangements of user cooperation tasks. In traditional groupware, when a user shares knowledge or information with one or more other users, the user is forced to input rigidly defined data. Our research

*Department of Computer Science, Ritsumeikan University
†UCHIDA YOKO CO.,LTD. Advanced Solution R&D Center
‡Department of Computer Science, Ritsumeikan University

solves this problem by providing a technique to flexibly describe the arrangement of user tasks.

Since our research recognizes context based on the exchanged description information, we call approach of such a this research "document based context aware computing".

2. Related Work

The paper [4] proposes the technique that the system generates context by making a plan based on a user's supplied goal, and binding its services to the context. This technique is similar to our own in that the system recognizes context based on the information that user's describe. However, our research is different in that the system can also use context history by using past arrangements of user tasks. Moreover, the purpose of our research differs by supporting tasks of two or more users.

The OKAR project [5] is supporting the productivity drive and knowledge creation in business. This project propose a format for describing the intellectual operating activity defined by OWL [6]. This project makes it possible to share operating activity information that is used in cooperation with a different system or apparatus, to accumulate business activity knowledge, and to utilize knowledge across organizations.

3. Proposed Technique

In this section, we describe the details of our proposed technique.

3.1 Outline

We focus on exchanging mail as the technique of passing context to the system by using that activity of the user's daily life. Because mail is using user supplied task arrangements and passing task modifications as well as results to other people in it by user in daily life. We believe that extending the mailer reduces the time when the user passes context explicitly. If the user can describe machine readable arrangements of user tasks by extending a mailer, the user can pass it to the system as well as other users who do joint work. Moreover, if the system can generate text from it, users can pass arrangements of user tasks in a format that other users easily understand. Users perform tasks by referring to arrangements of user tasks passed with the extended mailer, filling out the reply, and replying in the new machine readable description. Thus the user can add to task arrangement modifications and can inform the system of task achievements.

For example, Mr. Fu who is an office worker who might receive the following mail from his boss containing an arrangement of tasks in order to prepare for a meeting.

> To: Mr. Fu
> From: Your boss
>
> ---
>
> Hello, in preparation for the following meeting, please decide on an agenda, reserve a conference room, and create a report.I would like to see the report, so please send it to me.Also, please print it.

The arrangement of their tasks can be shown with a UML activity diagram where tasks are oval-shaped and task order relationships are shown with an arrow. Also, "report" related to the task can be shown with dashed lines.

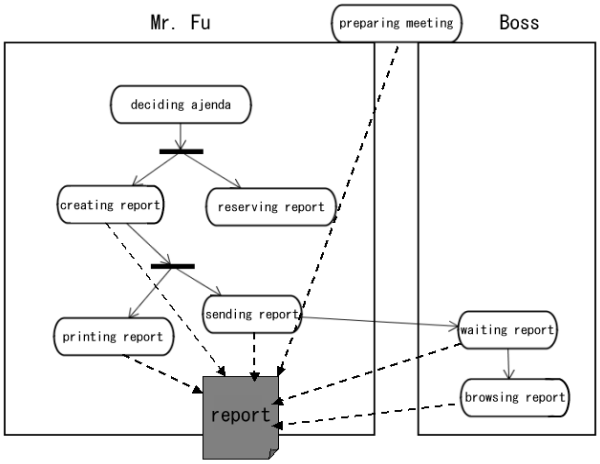

fig 1: Activity diagrams

This research enables the description of relationships such as in Fig. 1 in a machine readable format. Users decide to impose the arrangement of user tasks, and pass it beforehand to the system. Further, the user passes progress modifications and results to the system as well as which tasks that user is doing and which have been completed. This way, the system recognizes which tasks a user can face, namely the system knows the user's current context. In the earlier example of sending the machine readable task arrangement description to Mr. Fu, Mr. Fu's boss describe arrangement of their tasks and send it to Mr. Fu. Mr. Fu receives the mail and passes progress such modifications and results back to the system. Therefore, as Mr. Fu's context changes the context aware service become available.

After Mr. Fu decides on an ajenda, reserves a conference room, and passes achievement of these tasks to the system:

1. Mr. Fu sits down in front of his PC. Then, he is asked whether or not he wants to use the word processor service from the PC; to which he answers "yes". Finally, he is able to write his report using the word processor service.

2. Mr. Fu finishes creating the report and passes it to the system and leaves the office on some

errand. Around that time his boss is asked whether he wants to view the report on his PC and he answers "yes". Then his boss is able to see the report with a presentation service.

3. Mr. Fu returns to the office, and he is asked whether he wants to print the report from the printer near the entrance of the office. He answers "yes", and the printer prints the report.

In our proposed technique, a user describes the arrangement of their tasks in a machine readable format and passes it in advance to the system. Further, the user passes progress information such a modifications and results to the system. This research recognizes context history by these techniques and enables recognition of cooperative tasks between multiple users as well as context from considering previously executed tasks. In this research, we propose a technique where users describe arrangements of user tasks as the machine readable "TaskTemplate". Further, we propose a ContextStackFramework that recognizes context with the help of user input and binds the context to service. In this research, we are not concerned with we describing every arrangement of tasks in TaskTemplate and we are not necessarily forced to. For example, although people use a score in order to play music, people are not necessarily forced to play according to the score. TaskTemplate is like the score. It is important for the user to be able to perform tasks as they wish when using TaskTemplate and ContextStackFramework.

3.2 TaskTemplate

The requirements for TaskTemplate are shown below.

3.2.1 Enduser Documenting

Since TaskTemplate is described by user in daily life, TaskTemplate enables to write easily also by enduser. For this reason, we make the structure of TaskTemplate simple. Besides, TaskTemplate enables to create new TaskTemplate by combining existing TaskTemplate.

3.2.2 Order Relationship

According to mail from Mr.Fu's boss, Mr.Fu does "deciding ajenda" after doing "reserving a conference room" or "creating report". Thus, arrangement of user tasks has a defined order. There are two kinds of order relationships: those which can be performed in parallel, and those which must be performed sequentially. TaskTemplate needs to be able to represent both in order to give an order relationship to the arrangement of user tasks.

3.2.3 The Element Which Define The Task Itself

Task has the elements which defines the task itself. For example, the "creating report" task has the elements which defines itself such as "Mr. Fu (do)" and "report". Elements such a "report" may be shared by two or more users. TaskTemplate enables to express such an element which constitutes the task itself.

Although tasks are made up of elements, the elements may not be OK at a certain time. For example, although "report" is an element of "creating report", the "report" is not completed at the beginning of the task however after the user finishes the task. Also, although "report" an element of "printing report", "report" is not available until it is written by the user. So task elements may also not be available until earlier tasks are completed. In other words, the user is doing task is satisfied element. In this research, we define task c? as satisfying all the elements in the task. Therefore, a task is not completed unless all the elements in the task are completed. TaskTemplate provides a slot for placing in the elements related to task. Satisfaction and dissatisfaction of the element are expressed by whether or not the element is contained in the slot. We refer to a slot as a "plugging slot".

3.2.4 Level of Task

User classifies arrangement of user tasks by giving a level to the task. According to mail from Mr.Fu's boss, the "preparing meeting" task is located in a higher rank level than the other tasks. The other tasks are located in a lower rank level from the "preparing meeting" task. In order that classifies arrangement of user tasks and enables it to describe, TaskTemplate enables to express level of task. In this research, a task that is a lower level than another task is refered to as a subtask.

A slot shared between multiple tasks expresses a link between subtask slot and the slot of a higher level task. For example, the "report" slot in the "creating report" task is linked to the "report" slot in the high level "preparing meeting" task.

3.2.5 Semantics of Task

If the system can't understand the kind of task and slot, the system can't bind a context to the appropriate service. For this reason, we need to provide semantics for tasks and slots in TaskTemplate by using existing ontologies.

3.3 ContextStackFramework

The requirements for ContextStackFramework are shown below.

3.3.1 Comunication with Users

ContextStackFramework recognizes context based on TaskTemplate and context accompllshments? from users. ContextStackFramework allows to comunication with users through mail, web interface, etc. For example, ContextStackFramework can comunicate with a user through TaskTemplate attached to mail from the user.

3.3.2 Management of TaskTemplate

Generally, users have a lot of arrangements of user tasks. For this reason, ContextStackFramework needs to be able to manage more than one TaskTemplate.

TaskTemplate has tasks in which more than one user cooperate by the user. In order for ContextStackFramework to obtain context for every user, it must manages context for every user.

3.3.3 Expression of Context

TaskTemplate is described as a relationship of task ordering and levels. This characteristic is similar to how one describes functions in procedural programing language. Functions which a computer can execute are pushed onto the function stack in the execution management architecture of procedural source programs. Computer executes calculations in a function which was pushed onto the function stack, until the computer finishes exeuting the function. Besides, the functions whose calculations have finished are pop off the function stack, and the function is separated as the processing of a computer. In other words, the function stack contains functions which the computer can execute. For this reason, ContextStackFramework is likened to a function stack, and ContextStackFramework expresses context which a user can enter in a TaskTemplate by using a stack. In this research, we call this stack ContextStack. ContextStack is pushed context. Context pushed onto ContextStack is popped off when a user do the context and achieve the context. In TaskTemplate, the plugging of all of a task's slots express completion of the task. For this reason we make sure that the client can plug slot elements.

3.3.4 Context to Services Binding

ContextStackframework has functionality to bind contexts to services when users complete tasks at the request of clients. The function bind context to service based on the conditions for filling a slot, task of semantics or slot of semantics. For example, when "creating report" is the context, ContextStackFramework binds it to a word processor service in order to plug into "report". "

4. Design

4.1 Design of TaskTemplate

In this section, we explain the design of TaskTemplate.

4.1.1 Overall Structure

TaskTemplate is defined as on XML Schema [7] and on RDF Schema [8]. One task has one XML and one RDF in a TaskTemplate structure. In the RDF of the task and task slots are described the semantics. In the XML, the task described by the RDF is maintained and the relationship between the slots of the task and the slots of subtasks as well as the order relationship of subtasks are described.

4.1.2 XML Description

The XML description elements are shown below.

- task-template

  task-template elements has "using" and "subtask-template" as child elements. Further, it has "value" as an attribute. The "value" attribute is specifies the location where the RDF exists either by relative path or URL.

- using

  The using element enables the handling of the location where the XML of subtasks of the task specified by "task-template" as a variable. It has "value" and "name" as attributes. The value attribute is specified the location where RDF exists by relative path or URL. The name attribute is specifies the name of the variable corresponding to the value attribute.

- subtask-template

  The subtask-template element has "slotlink" and "prerequsite" as child elements. Further, it has "name" as an attributes. The name attribute specifies the optional variable name of subtask that specifies the using element.

- slotlink

  In order to that slotlink element enables us to express slots shared between tasks by expressing links between a task slot and a subtask slots specified by the task-template elements. The slotlink element has "slot" and "subslot" as attributes. The slot attribute specifies the URI of the task slot specified in the task-template element. The subslot attribute specifies the URI of the task slot specified in the subtask-template element.

- prerequisite

  The prerequisite element enables us to express orders relationship of subtasks. It has "name" as an attribute. The name attribute specifies the name variable of the subtask that executed before the subtask shown in "subtask-template".

4.1.3 RDF Description

In the RDF about task and task of slots are described. The described information includes: task semantics, task title and description, slot semantics as well as slot title and description. In order to ensure that the RDF description provides the ontolgy for expressing task and slot of semantics, it is defined with RDF.

The vocabulary of RDF description is defined as an RDF Schema. In this paper, task and slot name space is described as "task" and "slot" respectively.

- Task

  The task is expressed as an instance of the task:Task class. We describe the property which has this task class in the domain. The task:type property specifies semantics of task as class of gerund or class of noun. The task:topicID property specifies the unique id of the TaskTemplate. The unique id is expressed as a hash of the mail address of the user and a timestamp

at the time they pass a TaskTemplate to the ContextStackFramework. The task:title and the task:description property provide human readable a title and description, respectively. The task:slot property specifies the slot which defines the task itself.

- Slot

  The slot is expressed as an instance of the slot:Slot class. We describe the property which has this slot class in the domain. The slot:type property specifies slot semantics as noun class. For example, the slot should be plugged executor of task when class of slot:Executor is specified at the slot. We assume that the element plugged into the slot can be referenced by the URI. For example, user can be referenced by their mail address. For this reason, the slot:identifire property is specified by a URI when the element is plugged into the slot. Besides, it contains an empty string when an element is not plugged into the slot. The slot:date property specifies form of YYYY-MM-DD when a slot is plugged.

4.2 Design of ContextStackFramework

ContextStackFramework is composed of 3 modules. They are ContextServerConnector, ContextServer and ContextStackManager. In this subsection, we explain the design of ContextStackFramework.

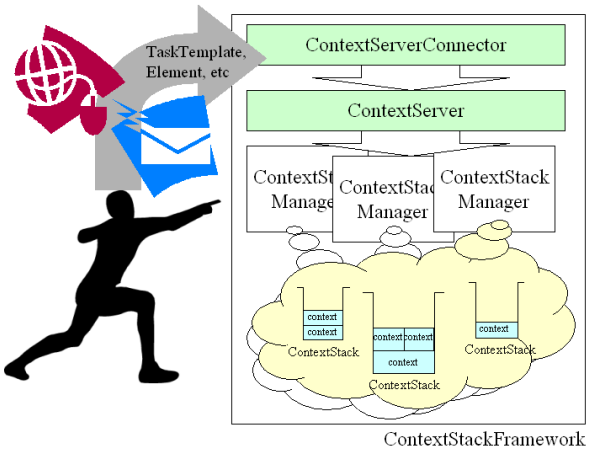

fig 2: ContextStackFramework

4.2.1 ContextServerConnector

The purpose of ContextServerConnector is to take full change of communication logic with clients. It leaves concrete logic to ContextServer as communication logic is made to be independent of concrete logic. For some reason, ContextServerConnector is defined as an interface, it can be accessed by using various communication techniques such as SOAP, RMI, etc depending on the implementation.

4.2.2 ContextServer

ContextServer performs the concrete logic that the ContextServerConnector receves from the client. Next, we describe it.

- Acceptance of TaskTemplate

  ContextServer accepts TaskTemplate that ContextServerConnector receves from the client based on a file or URIuri and then parse it. If TaskTemplate has tasks performed by multiple users, ContextServer parses the TaskTemplate for each user. And, ContextServer searches the ContextStackManager that manages TaskTemplate and contexts for every user transfers the TaskTemplate.

- Plugging Slot Element Acceptance

  ContextServer receives the element for plugging slot of optional user that ContextServerConnector receives from the client. Besides, ContextServer looks up ContextStackManager the user's through which it transfers the element to be plugged in ContextStackManager. At the time, there are 2 cases: the client transfers the element specifying the destination slot, and client transfers the element without specifying the slot.

- Context to Service Binding

  ContextServerConnector receives an optional user service list from the client and the ContextServer transfers it to user's ContextStackManager. A service is a table of a service id and a description of service ,and a service list is a list of services. Service description describes in the service used the kind of tasks and slots. ContextServer receives contexts from ContextStackManager and it binds contexts to services based on service description in the service list. And, ContextServer creates index tables of context and service id. Additionally, ContextServer transfers these index tables to requesting client via ContextServerConnector.

4.2.3 ContextStackManager

ContextServer exists for every user and ContextServer manages TaskTemplate received by ContextServerConnector. When ContextStackManager creates a ContextStack that manages context by using structure of stack and ContextStackManager pushed context based on level of task to ContextStack. After that, if a context is completed, ContextStackManager pops the context and it pushes the next context based on the task level and order relationship of tasks of orders and levels.

When ContextStackManager accepts an element for a plug in slot, ContextStackManager plugs the element in, if the slot is specified. When the plug in slot is not specified, a slot that it looks like

the element can be inserted into based on the element semantics is searched for, and the search result is returned to the requesting client via the ContextServerConnector. When the client receives the result, the client plugs the element in based on the result.

5. Current Status

Preparing TaskTemplate schema and implementing ContextStackFramework is finished currently. ContextSeverConnector is implemented as a service provider of in the UnitedSpaces [9] that is network service architecture.

UnitedSpaces in a logical space which manages services and users. Users can login to a space by using client software, obtain a service list for the space, and can then use services by selecting them. At such time, it is rack for user to select what kind of service.

By binding contexts to services and providing a service list we think that we efficiently select services based on the user. We implement mechanisms for this purpose. Unlike the example of the mail shown in section 3, we embedded the mechanisms for creating TaskTemplate in everyday software such as web browser, texteditor, etc. For example, we embedded the button for creating TaskTemplate in a web browser. When the button is pushed by a user who is loggined into space, the button creates TaskTemplate for the current web content. The button press also results in the passing of "browsing web content" TaskTemplate to ContextStackFramework through ContextServerConnector. For this reason, when user finds interesting web content, they push a button, and when the user logs into space that has web browser service, the user obtains service list that contains a service bound to "browsing web content" context. Thus, user can select and use services efficiently. However in implementations up to now, we cannot such as ease of TaskTemplate description, etc.

6. Conclusion and Future Work

In this paper we proposed a technique of recognizing contexts based on the concept of a user activity recognizing context and specifying it by describing an arrangement of everyday tasks in a machine readable format.

In section 6, in order to ensure that users can describe TaskTemplate we describe a mechanism for automatically creating TaskTemplate. However, this research also assumes users describe TaskTemplate during their daily activities such as example of mail in section 3.1. We believe that in order for the user to describe TaskTemplate, the user should be able to easily describe them. However, up to now, although we simplified the description of TaskTemplate by simplifying modeling of TaskTemplate, we cannot determine the complexity of user's describing the TaskTemplate in RDF/XML by hand.

In the future, we will examine GUI editor and intermediate language for that user can describe TaskTemplate. Besides, we will implement the mailer application shown in section 3.1 and we enables to easily describe TaskTemplate. Simultaneously, we will evaluate usability up until a user enjoys services and comprehensibility of the modeling of TaskTemplate.

References

[1] M.Weiser: "The Computer for the 21st Century," Sicentific American, September 1991,pp.94-100.

[2] Anind K. Dey, Daniel Salber and Gregory D. Abowd: "A Context-based Infrastructure for Smart Environments, In the Proceedings of the 1st International Workshop on Managing Interactions in Smart Environments (MANSE '99)," Dublin, Ireland, December 13-14, 1999. pp. 114-128.

[3] Albrecht Schmidt, Kofi Asante Aidoo, Antti Takaluoma,Urpo Tuomela, Kristof Van Laerhoven, Walter Van de Velde: "Advanced Interaction in Context, 1th International Symposium on Handheld and Ubiquitous Computing (HUC99)," pp. 89-101, 1999.

[4] Maja Vukovic: "Plan Based Application Modeling for Context Awareness" Doctoral Colloquium. The Sixth International Conference on Ubiquitous Computing (UbiComp). 2004, Nottingham, UK.

[5] Ontology for Knowledge Activity Resources, http://www.labs.fujitsu.com/jp/techinfo/okar/.

[6] OWL, http://www.w3.org/TR/owl-features/.

[7] XML Schema, http://www.w3.org/XML/Schema.

[8] RDF Schema, http://www.w3.org/TR/rdf-schema/.

[9] Yu Enokibori, Nobuhiko Nishio: "Realizing A Secure Federation of Multi-Institutional Service Systems," System Support for Ubiquitous Computing Workshop (UbiSys2004), Part of UbiComp2004, Nottingham, UK, Sept., 2004.