Workflow Language based on Web Services for Autonomic Services in Ubiquitous Computing

Joohyun Han, Eunhoe Kim, Jaeyoung Choi
School of Computing, Soongsil University, Seoul, Korea
{jhhan, ehkim}@ss.ssu.ac.kr, choi@ssu.ac.kr

Abstract

A workflow in ubiquitous computing environments must represent context information on transition constraints to support context-aware services. Moreover, it requires web service interfaces, which are independent of heterogeneous platforms, protocols, and languages. In this paper, we present uWDL, which is a workflow description language based on web services for ubiquitous computing and provides users with context-aware and autonomic services. Furthermore, we describe a structural context model to specify context information on transition constraints of uWDL.

Key words: Ubiquitous Computing, Workflow, Web services, Context-awareness

1. Introduction

Business and distributed services in traditional computing environments have a workflow process, or a flow of related tasks. The workflow supports service automation through a sequence of rules to process tasks. Users in ubiquitous environments may want to receive services in an appropriate form, in time, and without direct user intervention on dynamically changing environments [1,2]. Therefore the workflow must be expanded and adapted to satisfy these requirements.

The workflow in ubiquitous computing must meet several requirements. First, it must depend on context information sensed from the physical environments, which include much more dynamic and various information compared to traditional computing environments. Second, it must provide context-aware services automatically based on such sensed information. Third, platform- and language-independent standard service interfaces are needed to integrate, manage, and execute ubiquitous applications, which consist of heterogeneous protocols on various platforms. Therefore, a new workflow for ubiquitous environments is required to specify ubiquitous context information as constraints of state-transition in service flow. This requires web service interfaces, which are not only already specified and widely used, but independent of various platforms, protocols, and languages. In this paper, we propose uWDL, which is a workflow description language for ubiquitous computing based on web services. uWDL can be interpreted and executed in applications, and provide users with context-aware and autonomic services. Later, we describe a structural context model to specify context information on transition constraints of uWDL.

2. Structural Context Model

A context in ubiquitous computing environments can indicate any information which is used to characterize the situation of an entity [3]. A context-aware application uses context information and performs context-appropriate operations [2,3]. In ubiquitous environments, all services head for context-aware services to provide services appropriate for the user's situation. In order to provide a context-aware workflow service in ubiquitous environ-ments, an appropriate service is selected and executed based on the context information. Workflow services must be executed conditionally, concurrently, or repeatedly according to the conditions specified in the workflow. It is also required that workflow specify the constraints that influence the execution order of services such as start time, end time, deadline, and conditions.

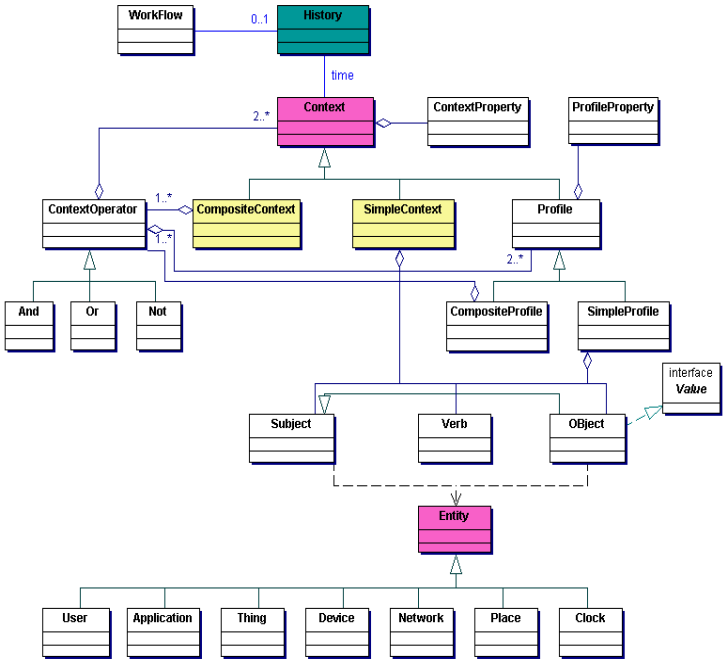

Fig. 1 The Structural Context Model

The goal of this paper is to specify ubiquitous context information on the state transition condition of workflow and enable the workflow to become context aware. We

propose the structural context model shown in Figure 1 to specify constraints affecting the execution order of services in ubiquitous workflow. The structural context model expresses ubiquitous context information from the viewpoint of knowledge structure. Because it has an information structure to express complex context information, it is possible to describe contexts in order to define constraints of state transition in uWDL language. Furthermore, because it is based on a single service interface which is a web service, it is not difficult to integrate, manage, and execute ubiquitous applications on various heterogeneous environments.

2.1 Simple Context and Composite Context

Context information is varied from a low level context describing facts in physical environments to a high level context that combines simple context information or processes by artificial intelligence. Ubiquitous context information might use ontology when it is implemented. Most ontology languages are based on RDF (Resource Description Framework). RDF is a language to describe the resource's meta-data and to express a resource as a tuple of 'subject-verb-objective.' So a simple context and profile information is described using an RDF expression, and complex context information is expressed using an ontology expression. The meaning of a verb used to describe a resource is varied by the context's attribute, and must be defined clearly using ontology. Context information could be expressed as a set of simple context information using the "subject-verb-object" form, which is called simple context information. Some context information cannot be expressed in 'subject-verb-object' form. Such context information could express a composite context, which combines a set of simple context information using the 'and', 'or', and 'not' operators.

3. A Workflow Language in Ubiquitous Computing

Current workflow languages specify data flows between services based on the web services. But these workflow languages [4,5] do not support the ability to select services using context, profile, and event information on ubiquitous computing environments. Therefore, it is difficult to express relationships among the services in ubiquitous environments using traditional workflow languages. Furthermore, middleware in ubiquitous computing which has heterogeneous features might be required to cover the heterogeneous environments.

uWDL is a ubiquitous workflow description language which specifies the relationship of service flows, characterized by the structural context model described in section 2. uWDL is a language that describes service flows and provides functionality to select an appropriate service based on high-level contexts, profiles, and events information which are collected by various sensors and structured by ontology.

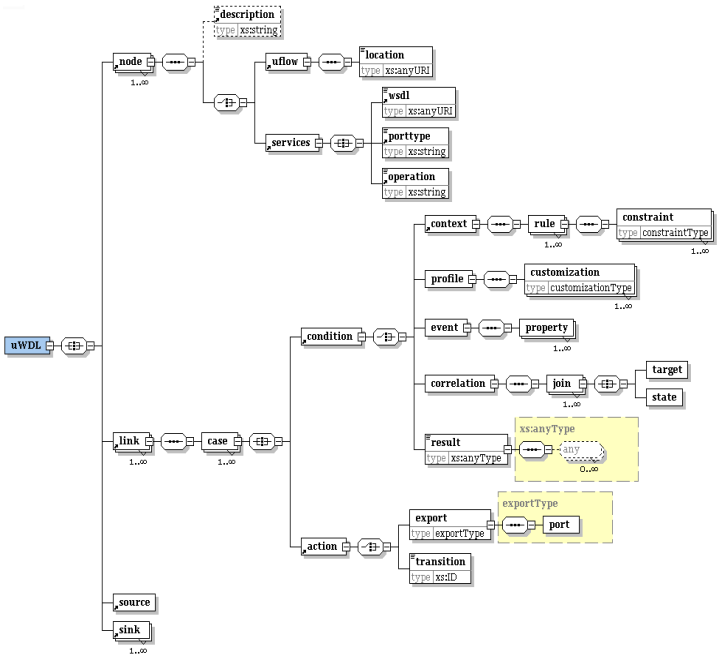

Fig. 2 uWDL Schema based on Structural Context Model

Figure 2 shows the schema structure of uWDL. uWDL provides the advantages of web service-based workflow language and also includes contexts, profiles and events to select service flows in ubiquitous environments.

3.1 <node> element

A BNF (Backus Naur Form) notation of the <node> element is as follows:

Table 1. A BNF Notation of the <node> Element

Node	:=	Description?,(Uflow | Services)
Uflow	:=	Location
Services	:=	Wsdl, Porttype, Operation

In Table 1, the <node> element points to one operation that provides a functionality of web services in ubiquitous environments. Web services use WSDL (Web Services Definition Language) to describe the port types and operations of specific web services. So, uWDL use the <services> element and sub elements of <services> - <wsdl>, <porttype>, and <operation> - to describe the service location, type, and operation of a specific web service. And the <uflow> element points a reference to another uWDL document.

3.2 <link> element

In Table 2, the <link> element is the most important part of the uWDL. It specifies context, profile, and event, aggregated from ubiquitous environments and defines the flow of services. The <link> element is composed of <condition> and the <action> elements. The <condition> element uses <context>, <profile>, and <event> sub elements to specify the context, profile, and event status

of a specific node, respectively. If the evaluated value of the status satisfies a given condition, the action described in the <action> element is performed. The <action> element consists of <export> and the <transition> elements, where <export> has a control link and a data link according to its attribute, and <transition> specifies the state change of a current node.

Table 2. A BNF Notation of the <link> Element

```
Link      := Case+
Case      := Condition, Action
Condition:= Context|Profile|Event|Correlation|Result
Context  := Rule+
Rule      := Constraint+
Profile   := Customization+
Event     := Property+
Correlation:= Join+
Join      := Target, State
Action    := Export | Transition
Export    := Port
```

The <condition> element makes a decision to select a proper service by context, profile, and event information. The most important element is the <context> element, which contains the <constraint> element to specify high-level context information generated by ontology and/or inference service in a form of the structural context model. The <constraint> element has the attributes of subject, verb, and object. For example, if a constraint is "temperature is higher than 30 Celsius," the expression of the constraint is "<constraint subject="Temperature" verb=">" object="30">". Each constraint of subject and object is provided by an entity of the structural context model. The entity indicates where the information of subject and all objects could occur in ubiquitous environments. Therefore the <constraint> element expresses a context by the relationship of the object and the subject, which are instances of the entity. The type attribute of the <constraint> element has an attribute of 'and', 'or', and 'not'. By using these types of attributes, it is possible to express the relationship among the simple contexts and therefore describe a high-level complex context. The <rule> element is a set of the <constraint> elements, and represents the high-level expression to decide a social situation. The <profile> element is required to express the user property. Profile information is the explicit information on an entity. Context information is obtained by sensing the environment and varies dynamically over time, while profile information is fixed or varies slowly. The <profile> element could express more formal context information than the <context> element. The <event> element is used to process event information generated by currently performing service and is expressed as changeOfLocation, changeOfProfile, and so on. The <profile> element also contains the <correlation> element to collaborate with other services.

The uWDL is able to integrate, manage, and execute various and heterogeneous services in ubiquitous computing environments because it is based on web services. It also specifies context information on the constraints of state-transition information by using the structural context model and as a result supports context-aware services. The structured context model also expresses ubiquitous status information in a structural viewpoint and could further describe complex context information. In the future, we will continue our research on the ubiquitous workflow engine which interprets and executes the uWDL document and on context infrastructure in order to provide context information in real-time to the ubiquitous workflow engine.

4. Scenario

The above-mentioned uWDL could determine a service by context and profile information. Because it is a language based on web services, the service could be used independently of protocols and platforms. To verify these properties, we present a scenario to prepare an office meeting using uWDL in ubiquitous environments. The scenario is as follows: "Implementing a service which prepares an office meeting automatically according to a schedule." A scenario developer conceptualizes the scenario as shown in Figure 3, and designs a service flow using the uWDL scenario editor.

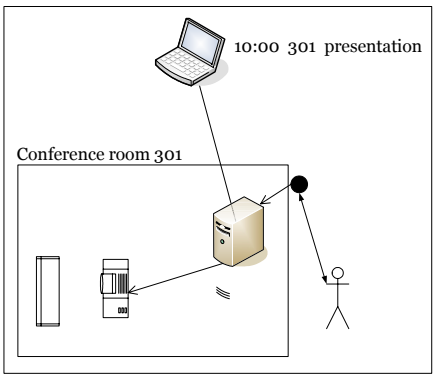

Fig. 3 Office meeting preparation scenario

At that time, the developer must decide which service is selected within the context and profile information. This process is done by describing context and profile information using the structural context model. The result is converted into a uWDL file parsed by the uWDL parser, and then executed. The scenario is as follows: "Ann records an appointment on her notebook computer that there is a presentation in Room 301 at 13:00 PM. Ann moved to Room 301 to participate in the meeting at 12:40 PM. There is a RFID sensor above room 301's door, and Ann's basic context information (such as her name, her notebook's IP address) is transmitted to a server. If the conditions, such as current time, situation, and user location, are satisfied, then the server downloads Ann's presentation file and executes a presentation program."

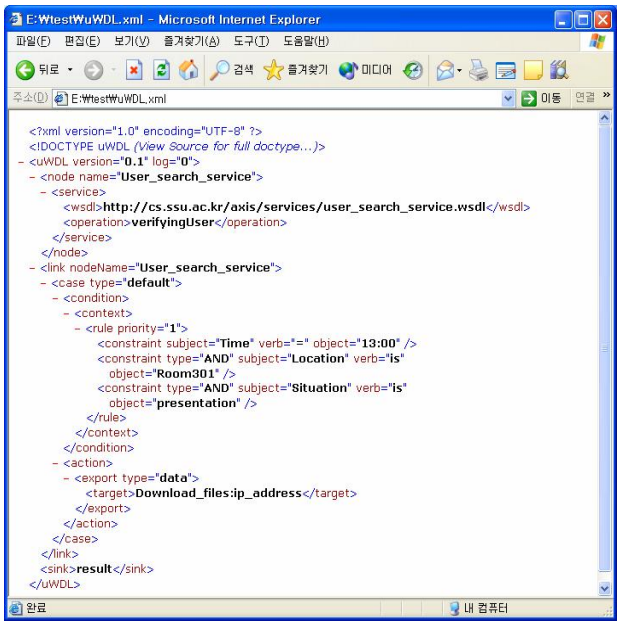

Fig. 4 XML Instance File

Figure 4 shows an XML instance file created using the uWDL scenario editor. This uWDL document is analyzed and executed by uWDLlexer and uWDLpaser. A uWDL document edited by a service scenario developer is scanned by the uWDLlexer line by line in string form. The uWDLlexer calls the Yytoken function to acquire a token value from the scanned token. A uWDL scenario document is divided as units of token by the uWDLlexer. The tokens are then parsed by uWDLParser which is a recursive descendant parser using the uWDL's DTD definition. The parsed scenario document is compared with a DTD tree and only executed by the service engine when the compared value is valid. Through this, we prove the uWDL document is a well-formed document. Finally, a service which is an instance of the uWDL application is executed by a workflow engine.

6. Related Work

There is much research being done to develop a context-aware application in ubiquitous environments. Many context-aware systems such as PARCTAB, Aura [6], and One.world define context-triggered action using if-then rules. At run-time, those systems provide context-aware services by executing context-triggered actions supported by event infrastructure based on a publish-subscribe mechanism. However, such research is mainly aimed at supporting sensibility of the user's context within one service and do not support flows of services. Gaia [7] is a CORBA-based ubiquitous middleware and integrates the distributed devices of a physical environment into a space called "Active Space." Furthermore, it enables communication among ubiquitous applications by exchanging context information. Because Gaia is implemented based on CORBA, it is not language independent. LuaOrb, Gaia's scripting language, provides functionality to the program and describes ubiquitous context information. However, it is difficult to express dependency, order, and concurrency, since initialization and instantiation of a program should be described sequentially. And as LuaOrb is not defined in an XML language, it is also difficult to provide compatibility with other middleware and to expand scripts. The uWDL (ubiquitous workflow description language) reflects the advantages of current workflow languages such as BPEL4WS, WSFL [4], and XLANG [5] and furthermore can describe context, profile, and event information as constraints in state-transition of services.

5. Conclusion

In this paper, we proposed a structural context model to specify context information on transition constraints of workflow in ubiquitous environments. Also we presented a ubiquitous workflow description language, uWDL, to specify service flows based on a structural context model where appropriate services are selected and executed concurrently or repeatedly, and to specify context-aware state transition which influences service flow at run time. Because the uWDL describes services using the web services and specifies service flows using the workflow, it is effectively able to be integrated, managed, and executed in ubiquitous applications, which consist of heterogeneous protocols on various platforms. Also, it can easily represent a scenario in ubiquitous computing, for it specifies ubiquitous context information as constraints of state-transition in service flows. In the future, we will continue our research on the semantic web and DAML+OIL to adapt them in our structural context model. We plan to develop the ubiquitous workflow engine which interprets and executes the uWDL document, and context infrastructure to provide context information in real-time to the ubiquitous workflow engine.

References

1. D. Saha, A. Mukherjee, "Pervasive Computing: A Paradigm for the 21st Century", IEEE Computer, IEEE Computer Society Press 25-31, 2003.
2. Guanling Chen, David Kotz, "A Survey of Context-Aware Mobile Computing Research", Technical Report, TR200381, Dartmouth College, 2000.
flow Management Coalition (1994)
3. Anind k. Dey, "Understanding and Using Context, Personal and Ubiquitous Computing", Vol 5, Issue 1, 2001.
4. Frank Leymann, "Web Services Flow Language (WSFL 1.0)", IBM, 2001.
5. Satish Thatte, "XLANG Web Services for Business Process Design", Microsoft Corp., 2001.
6. D. Garlan, D. Siewiorek, A. Smailagic, and P. Steenkiste, "Project Aura: Towards Distraction-Free Pervasive Computing", IEEE Pervasive Computing, 2002.
7. Manuel Roman, Christopher K. Hess, Renato Cerqueira, Anand Ranganathan, Roy H. Campbell, Klara Nahrstedt, "Gaia: A Middleware Infrastucture to Enable Active Spaces", IEEE Pervasive Computing, 74-83, 2002.