A Ubiquitous Workflow Service Framework

Joohyun Han, Yongyun Cho, Eunhoe Kim, and Jaeyoung Choi

School of Computing, Soongsil University,
1-1 Sangdo-dong, Dongjak-gu, Seoul 156–743, Korea
{jhhan, yycho, ehkim}@ss.ssu.ac.kr, choi@ssu.ac.kr

Abstract. In ubiquitous environments, all services head for context-awareness to provide appropriate services for a user's situation. However, it is hard to implement all kinds of things related to context managements. In this paper we propose a ubiquitous workflow service framework, named uFlow, based on a structural context model and uWDL, which is a ubiquitous workflow description language. Service developers can easily describe context-aware services using the uFlow framework so long as they only select available services based on Web Services and describe context information as a transition condition of workflow services. In order to verify the effectiveness of the uFlow framework, we designed and implemented a service scenario described with uWDL, and demonstrated that the scenario provides users with appropriate services according to a user's situation in ubiquitous computing environments.

1 Introduction

WfMC (Workflow Management Coalition) states that a workflow expresses flows of subtasks until a process is completed using standardized methods [1]. Between the subtasks in a workflow, there exist various relationships such as dependency, ordering, and concurrency. Workflows describe flows of subtasks using a workflow language. A workflow management system manages and controls flows of subtasks using state-transition constraints specified in the workflow language. Modeling a workflow can help software designers better understand how to support users when they design applications.

An application or a service that uses context information or performs context-appropriate operations is called a context-aware application or a context-aware service [2, 3]. In order to provide a context-aware service in ubiquitous environments, an appropriate service is selected and executed based on context information. Service developers should describe and handle context information to build context-aware services. However, it is so difficult to implement all sorts of things related to context managements such as context wrapper, context query system, ontology server, and so on.

In this paper, we propose a ubiquitous workflow service framework, named uFlow, based on a structural context model and uWDL. uWDL is a ubiquitous workflow description language and it can specify the context information on the transition conditions of workflow services to provide users with adaptive

M. Gavrilova et al. (Eds.): ICCSA 2006, LNCS 3983, pp. 30–39, 2006.

services for a user's current situation, and the structural context model is used to express context information in uWDL. Service developers can easily design context-aware services using the uFlow framework so long as they only select available services based on Web Services and describe context information as a transition condition of workflow services.

2 Related Work

Gaia [4] supports a service environment in which ubiquitous applications can communicate context information to each other. It depends on a specific protocol which is not widely used because it is based on CORBA middleware. LuaOrb, that is Gaia's script language, can instantiate applications and interact with execution nodes to create components and easily glue them together, but it can't express dependency or parallelism among the services because it describes only a sequential flow of specific services. The uFlow framework is a workflow system based on Web Services which are platform- and language-independent standard service interfaces, so it can express dependency and parallel execution among the services in heterogeneous ubiquitous computing environments.

BPEL4WS [5], WSFL [6], and XLANG [7] are Web Service-based workflow languages for business processes and distributed computing environments. They support service transition, and use XML-typed messages defined in other services using XPath. Context information is a complex data set that includes data types, values, and relations among the data types. XPath cannot sufficiently describe diverse context information because it can use only condition and relation operators to decide transition conditions. uWDL uses a context triplet - subject, verb, and object - in order to express high-level context information as transition conditions, which can not be supported by existing workflow languages.

3 The Key Components for uFlow

A context in ubiquitous computing environments indicates any information that is used to characterize the situation of an entity [3]. In ubiquitous environments, all services head for context-aware services to provide appropriate services for a user's situation. In order to provide context-aware workflow services in ubiquitous environments, an appropriate service is selected and executed based on context information. Therefore we designed two components, which are a structural context model and a ubiquitous workflow description language to use the context information as the constraints of the state transition in ubiquitous workflow services. The two components are designed based on knowledge structure to express the context information in a simple and flexible way.

3.1 Structural Context Model

A structural context model expresses ubiquitous context information from a viewpoint of knowledge structure. Because it has an information structure to

express complex context information, it is possible to describe contexts to specify the context information on a transition condition of services in uFlow scenario documents. Figure 1 shows a class diagram of the structural context model. Context information can be any information that describes a situation of an entity. An entity is a person, place, physical, or logical thing which is considered in ubiquitous computing environment. We designed the structural context model which is an ontology-based context model for uFlow using OWL (Web Ontology Language), and it describes context information as entities having context types which have its values. The ontology language OWL builds on RDF (Resource Description Framework) [8], and a RDF statement is always a triple of resource, property and value, in that order. Because our model is based on ontology using OWL, our core concepts which are subject, verb, and object can be mapped into resource, property, and value on RDF, respectively.

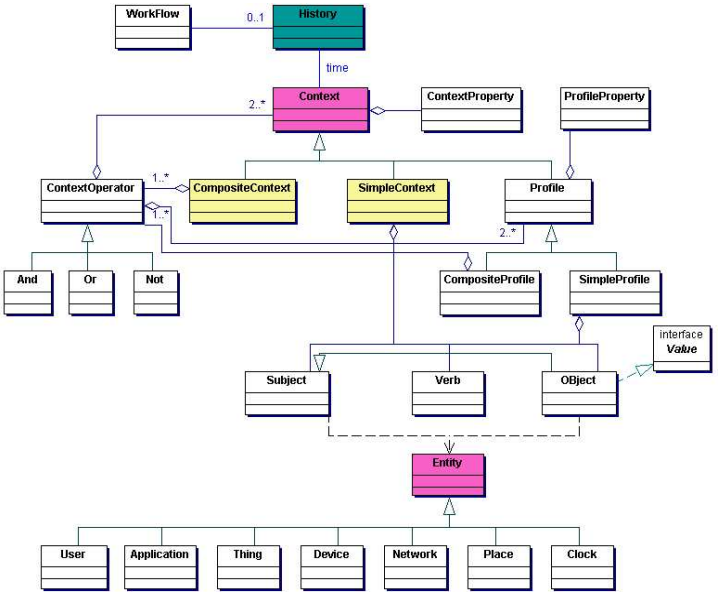

Fig. 1. Structural context model

3.2 Ubiquitous Workflow Description Language

A workflow management system manages and controls flows of subtasks using state-transition constraints specified in a workflow language. Although current workflow languages such as BPEL4WS, WSFL, and XLANG can specify the flows among services based on Web services, these workflow languages do not support the ability that controls the state-transition constraints using context, profile, or event information in ubiquitous computing environments. uWDL (Ubiquitous Workflow Description Language)[9] is a Web Services-based workflow language that describes service flows, and provides the functionalities to

select an appropriate service based on high-level contexts, profiles, and events, which are obtained from various sources and structured by Ontology [10]. To provide these functionalities, uWDL specifies context and/or profile information as a triplet of {subject, verb, object} based on the structural context model for rule-based reasoning which can effectively represent the situation in a simple and flexible way. Figure 2 shows the schema structure of uWDL.

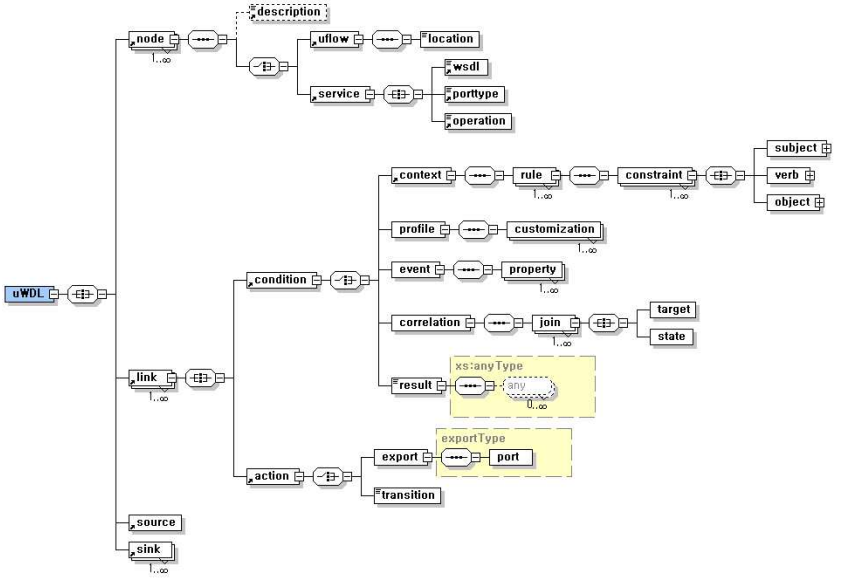

Fig. 2. uWDL schema

4 Ubiquitous Workflow Service Framework

Service developers have to describe and manipulate context information for context-awareness of services. However, they have a great difficulty in implementing all kinds of things related to context managements such as context wrapper, context query system, ontology server, and so on. We propose a ubiquitous workflow service framework named uFlow based on the key components in Section 3. Service developers can easily design context-aware services using the uFlow framework so long as they only select available services based on Web Services and describe context information as a transition condition of workflow services. The uFlow framework consists of uFlow scenario editor, uFlow engine, and uFlow context processor as shown in Figure 3.

A service developer can use the uFlow scenario editor to create a scenario document written with uWDL. The service developer can query context information to the ontology server through context browser to obtain standard vocabularies of context information for a specific domain and specifies the context information as a transition condition of workflow services. The scenario document created

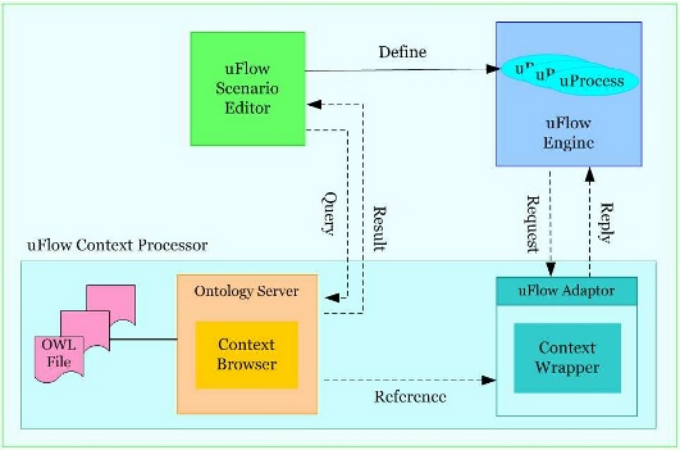

Fig. 3. uFlow framework

by uFlow scenario editor is delivered into uFlow engine in order to parse and manipulate the context information according to the service flows. The uFlow engine collects context information through a context wrapper in uFlow context processor and compares these context information with the context information described in the scenario document. If the matching result is true, an appropriate service is executed by the uFlow engine. The detailed explanations are as follows.

4.1 uFlow Scenario Editor

uFlow scenario editor is a tool for developers to easily design scenario documents without detailed understanding of uWDL schema. Developers can select currently available services based on Web Services and describe context information as a transition condition of the services. The uFlow scenario editor provides drag and drop capabilities to <node> and <link> elements in uWDL and consists of available services, element explorer, and context information obtained from current sensing environments. A scenario document is created by uflow scenario editor, and translated and executed by uFlow engine. Figure 4 shows uFlow scenario editor.

4.2 uFlow Engine

A uWDL document designed for a specific scenario should be translated and executed to provide adaptive services for a user's situation. For this purpose, we need a process to manipulate contexts aggregated from a sensor network. Figure 5 shows uFlow engine for handling context information expressed in uWDL. uWDL parser parses a uWDL scenario document and produces a DIAST (Document Instance Abstract Syntax Tree) [11] as a result. A DIAST represents the syntax of a scenario document, and is used to compare contexts expressed

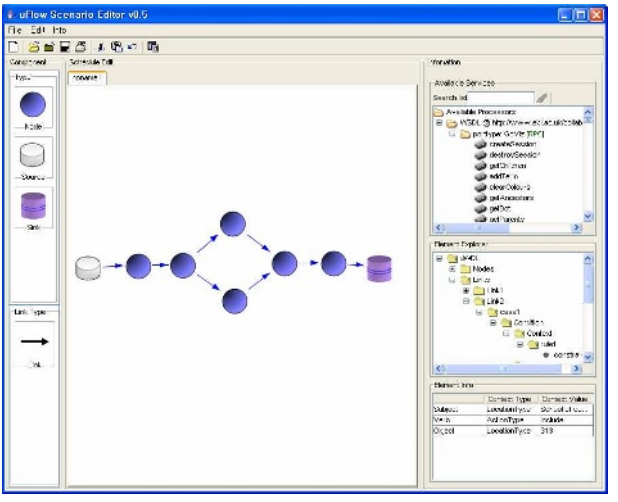

Fig. 4. uFlow scenario editor

in a scenario with entities aggregated from a sensor network to verify their coincidence. A context is described by one or more constraint elements, and each constraint is represented by a context triplet of {subject, verb, object} in a sequence. In Figure 5, a partial subtree in dotted lines indicates a subtree that makes up context constraints in the scenario.

A context mapper extracts types and values from objectified entities aggregated from a sensor network, and composes a subtree which consists of subject, verb, and object information. It then compares the type and the value of an entity with those of the constraint element in the DIAST subtree, respectively. If the type of the entity matches with its counterpart in the constraint element, the context mapper regards it as a correct subelement of the constraint element. If each entity has the same type, it may be ambiguous to decide the context's constraint according to its entity type only. The problem can be resolved by comparing the value of the objectified entity with that of the constraint element in the DIAST subtree.

4.3 uFlow Context Processor

uFlow context processor takes a responsibility of providing context information with uFlow scenario editor and uFlow engine. uFlow context processor consists of context browser, context wrapper and ontology server. Usually a Context infrastructure treats low-level context information that is raw contextual information which comes from sensors such as temperature, noise level, and location. However, uFlow needs not only low-level context information but also high-level context information that is combined by two or more low-level context information. Ontology server has functions which reason high-level context information from some low-level context information, and which reason explicit

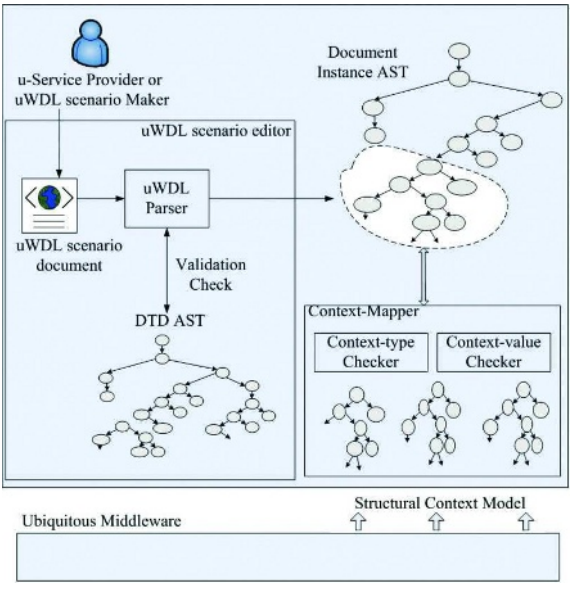

Fig. 5. uFlow engine

context information from implicit context information through our Ontology-based context model and ontology reasoner.

Context wrapper transforms context information obtained from a sensor network into a form of structural context model adequate to uFlow Engine. Context information structured using the structural context model consists of a enity(subject), a type of the entity(verb), and a type of the value(object). uFlow scenario editor is a tool which defines a sequence of services using context information. uFlow scenario editor requests context browser to browse available context information provided by ontology server, and context browser delivers them to uFlow scenario editor.

5 Experiments

In this section, we show a process to decide a state transition condition according to context information. For testing, we simulated a ubiquitous office in Figure 6 using uFlow framework. The purpose is "implementing a service which prepares an office work automatically according to a user's situation." Context information is simulated in a virtual office environment based on GUI according to a variation on the schedule, time, and/or user's location and preference. These context information is structured using the structural context model in Section 3 and transmitted to uFlow engine in order to identify current situation. uFlow engine executes related services which exist in a form of Web Services according to the user's situation. The scenario context tab in Figure 6 shows the progress of uFlow engine how to select a service according to dynamically incoming context information.

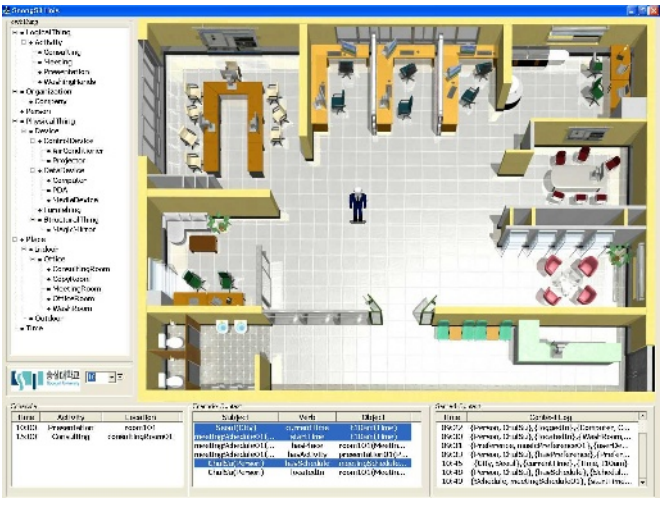

Fig. 6. The Simulation of a Ubiquitous Office

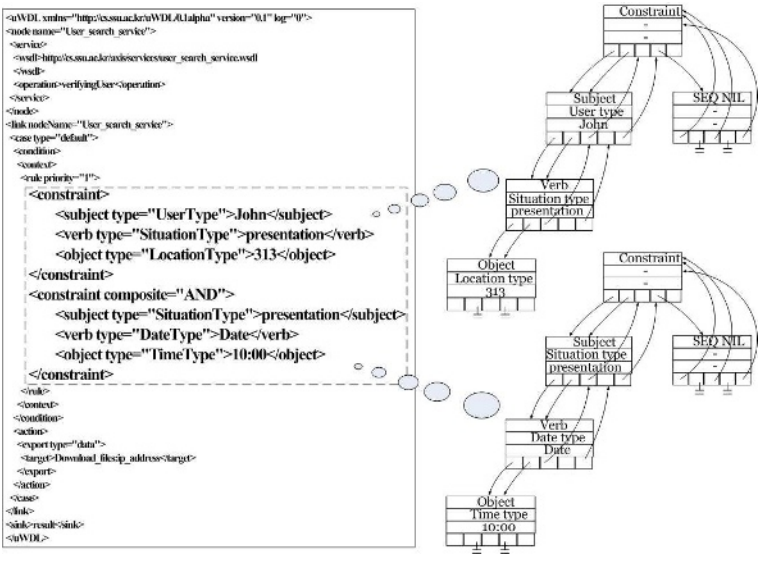

Fig. 7. A scenario document and a DIAST's subtree produced by uWDL parser

Figure 7 shows a scenario document designed using the uFlow scenario editor. If uFlow engine receives context data objectified as (SituationType, presentation), (UserType, Michael), (UserType, John), and (LocationType, 313), it compares the contexts' types and values with the subtree's elements. In this case, the context (UserType, Michael) is not suitable for anywhere in the subtree's elements, so uFlow engine removes the context. For the experiment, we named a

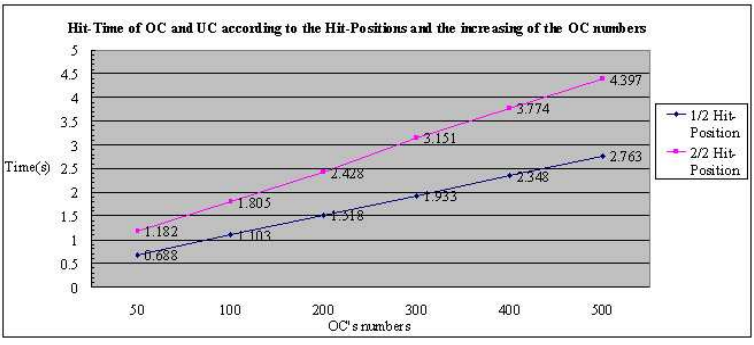

Fig. 8. A hit-time for hit-position and the number of OCs

context of a scenario document as UC(uWDL Context) and a context obtained from a sensor network in ubiquitous computing environments as OC(Objectified Context). uFlow engine decides a service transition through a comparison between UCs and OCs. A context described in the scenario document consists of a limited number of UCs. On the other hand, contexts obtained from a sensor network can be produced as innumerable OCs according to a user's situation. Therefore, uFlow engine should select quickly and correctly an OC coinciding with a UC from such innumerable OCs.

In Figure 8, we generated a lot of OCs incrementally, and measured how fast the suggested uFlow engine found the OC of the produced OCs that coincided with the UCs of the scenario document shown in Figure 7. To get the hit-time, we placed the OCs coinciding with the UCs in the middle and the end of the OCs that we produced randomly. We used a Pentium 4 2.0 GHz computer with 512MB memory based on Windows XP OS for the experiment. We increased the OC's amounts by 50, 100, 200, 300, 400, and 500 incrementally.

In Figure 8, 1/2 hit-position means the position of the OC coinciding with the UC is the middle of the randomly produced OCs, and 2/2 hit-position means the position of the OC is the end of the randomly produced OCs. As shown in the result, the hit-time is not increased greatly regardless of the OCs's considerable increase. This result shows that the suggested uFlow engine can sufficiently support context-aware services.

6 Conclusion

In this paper, we proposed a uFlow framework for ubiquitous computing environments. The uFlow framework was designed based on uWDL which can easily describe service flows and the structural context model to express context information in uWDL. uWDL can specify the context information on transition constraints of a service workflow in ubiquitous computing environments, and is designed based on Web services. The uFlow framework consists of uFlow scenario editor, uFlow engine, and uFlow context processor. It is able to integrate, manage, and execute various heterogeneous services in ubiquitous environments.

Therefore, uFlow framework provides users with appropriate services according to the user's context information. We developed a scenario described with uWDL, and we demonstrated that the uFlow framework can provide users with autonomic services in ubiquitous computing environments. In the near future, we will expand uWDL schema to express more detailed situations by assigning semantic information to Web services.

Acknowledgements

This work was supported by the Ubiquitous Autonomic Computing and Network Project, funded by the Korean Ministry of Information and Communication (MIC).

References

1. D. Hollingsworth: The Workflow Reference Model. Technical Report. TC00–1003. Workflow Management Coalition (1994)
2. Guanling Chen, David Kotz: A Survey of Context-Aware Mobile Computing Research, Technical Report, TR200381, Dartmouth College (2000)
3. Anind k. Dey: Understanding and Using Context, Personal and Ubiquitous Computing. Vol 5. Issue 1. (2001)
4. Manuel, Roman, Christopher, K.: Gaia: A Middleware Infrastructure to Enable Active Spaces. IEEE Pervasive Computing (2002) 74–83
5. Tony, Andrews, Francisco, Curbera, et al.: Business Process Execution Language for Web Services. BEA Systems. Microsoft Corp. IBM Corp., Version 1.1 (2003)
6. Frank, Leymann: Web Services Flow Language (WSFL 1.0), IBM (2001)
7. Satish, Thatte: XLANG Web Services for Business Process Design, Microsoft Corp. (2001)
8. W3C: RDF/XML Syntax Specification, W3C Recommendation (2004)
9. Joohyun Han, Yongyun Cho, Jaeyoung Choi: Context-Aware Workflow Language based on Web Services for Ubiquitous Computing, ICCSA 2005, LNCS 3481, pp.1008–1017, (2005)
10. Deborah, L., McGuinness, Frank, van, Harmelen, (eds.): OWL Web Ontology Language Overview, W3C Recommendation (2004)
11. Aho, A., V., Sethi, R., Ullman, J., D.: Compilers: Principles, Techniques and Tools. Addison–Wesley (1986)