A uWDL Handler for Context-Aware Workflow Services in Ubiquitous Computing Environments

Yongyun Cho, Joohyun Han, Jaeyoung Choi, and Chae-Woo Yoo

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

Abstract. To develop context-aware workflow services in ubiquitous computing environments, a service developer must describe and recognize context information as transition constraints. uWDL (ubiquitous Workflow Description Language)[1] is a workflow language that describes the situation information of ubiquitous environments as a rule-based service transition condition. In this paper, we suggest a uWDL handler that supports workflow's service transition to be aware of user's condition information. The uWDL handler consists of a uWDL parser and a uWDL context mapper. The uWDL parser represents contexts described in the scenario with sub-trees of a DIAST (Document Instance Abstract Syntax Tree) as a result of the parsing. To derive the right transition of workflow services, the uWDL context mapper compares contexts described in subtrees of DIAST with a user's situation information generated from ubiquitous environments by using a context comparison algorithm. Therefore, the uWDL handler will be used in developing context-aware workflow applications that can change the flow of a service scenario according to the user's situation information in the ubiquitous computing environment.

1 Introduction

Ubiquitous computing environments mean that a user can connect with a network freely and receive services that he wants, anyplace and anytime [2, **?**]. A workflow model for business services in traditional distributed computing environments [3] can be applied as a service model to connect services related in ubiquitous computing environments and express service flows [1]. However, a workflow in ubiquitous computing environments must decide a service transition according to the user's situation information that is inputted dynamically [2]. For that, a workflow language for ubiquitous environments must be able to express the user's situation information as service transition conditions in a workflow service scenario. uWDL (ubiquitous Workflow Definition Language) is a workflow language based on a structural context model which expresses context information as transition constraints of workflow services [1, **?**]. Through a workflow service scenario document in uWDL, developers can represent context information as workflow state transition constraints in order to support a context-aware service transition of workflows. To develop application programs

T. Enokido et al. (Eds.): EUC Workshops 2005, LNCS 3823, pp. 131–140, 2005.

with a workflow language, a developer commonly needs a handler that processes a document written in that language and interprets the structure and the meaning of it.

In this paper, we present a uWDL handler that verifies the validation of a uWDL workflow service scenario document and derives the service transition according to a user's state information being inputted dynamically in ubiquitous environments. For that, the uWDL handler consists of a uWDL mapper and a uWDL parser. The uWDL parser parses a uWDL scenario document, and produces DIAST (Document Instance Abstract Syntax Tree), which represents the document's structure information. To decide a workflow service transition, a UWDL mapper compares contexts described in DIAST with the user's situation information offered from a sensor network.

2 Related Work

2.1 Context-Aware Workflow and Workflow Language

Context in a ubiquitous environment means any information that can be used to characterize the situation of an entity [3]. An application or system that uses context information or performs context-appropriate operations is called a context-aware application or context-aware system [4, **?**]. Ubiquitous workflow is dependent on context information that is sensed from the physical environment, and provides a context-aware service automatically based on that sensed information. The ubiquitous workflow is required to specify ubiquitous context information as state-transition constraints. The existing workflow languages, such as BPEL4WS [5], WSFL [6], and XLANG [7], are suitable for business and distributed computing environments. These languages use the results of the former services and the event information of services as transition conditions of services. However, they do not include any elements to describe context in ubiquitous computing environments to workflow services. For example, XPath is unsuitable for expressing high-level situation information that comes from ubiquitous environments, because it has only logic and condition operators.

2.2 uWDL (Ubiquitous Workflow Description Language)

uWDL [1] can describe context information as transition conditions of services through the <context> element consisting of the knowledge-based triple entity - subject, verb, and object. The uWDL reflects the advantages of current workflow languages such as BPEL4WS, WSFL, and XLANG, and also contains rule-based expressions to interface with the DAML+OIL [8] ontology language. In uWDL, a simple context and profile information are described using an RDF expression [9], and complex context information is expressed using an ontology expression. Figure 1 shows uWDL's schema.

In Figure 1, the <node> element points to Web services in ubiquitous environments and it conforms to a web service's operation. The <link> element contains

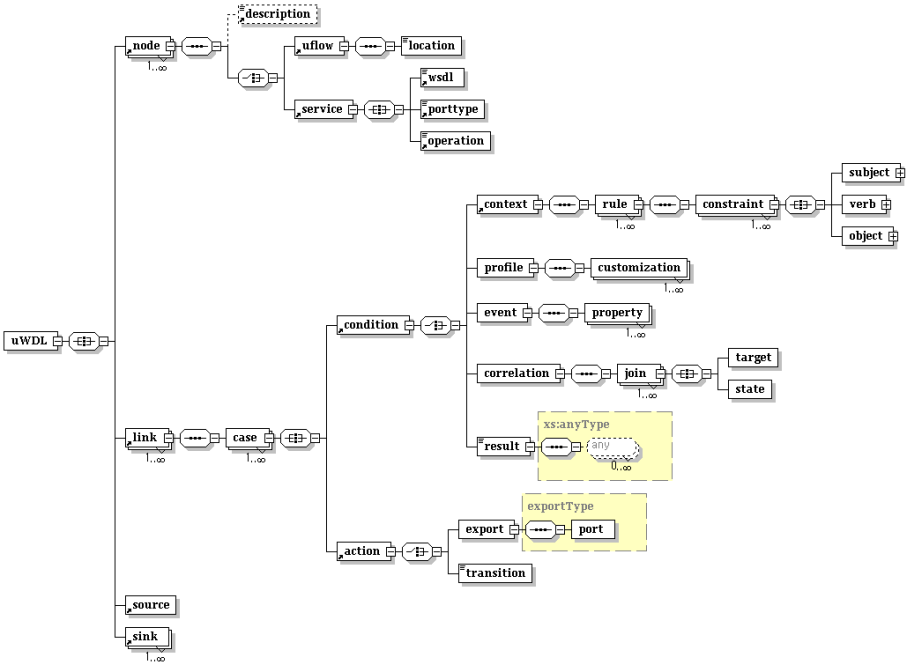

Fig. 1. uWDL Schema

a sub element <condition>, which selects the service flows. The <context> element contains the <constraint> element in order to specify high-level context information generated by ontology and inference services as a form of structural description. The <constraint> element has the triplet sub-element of <subject>, <verb>, and <object> based in RDF. The <node> element points to one operation that provides a functionality of Web services in ubiquitous environments. The <transition> element specifies the state change of a current node. The <condition> element makes a decision to select a proper service by context, profile, and event information. The composite attribute of the <constraint> element has a value of 'and', 'or', and 'not'. By using these attributes, we can express the relationship between simple contexts and describe a high-level complex context. The <rule> element means a set of the <constraint> elements.

2.3 Context Parser

To develop an application program in a context definition language, a developer needs a parser to parse the structure and the meaning of a document that is made out in the language. Jena2 [11] is useful as a parser for such ontology languages as RDF, DAML+OIL, and OWL [12] to define context. Jena2 parses ontology based in existent RDF as well as contexts of DAML+OIL, OWL, N3, DB, and so on. Jena2 redefines low-level contexts from a sensor network in

ubiquitous environments as high-level contexts based in RDF. However, Jena2 is not suitable for processing workflow scenario documents like a uWDL scenario document that includes the <context> element to describe contexts. Therefore, developers may require a parser that can recognize and parse workflow service scenario documents describing contexts like a uWDL workflow service scenario document in ubiquitous computing environments.

In this paper, we design and implement a uWDL parser that parses the structure and meaning of a uWDL document. Also, we propose an algorithm for comparing contexts inputted in RDF-based triple form in ubiquitous environments with contexts described in uWDL workflow service scenario documents. Through the algorithm, the uWDL mapper makes the workflow perform context-aware service transitions according to a user's situation.

3 A uWDL Handler

3.1 A System Architecture

In this paper, we propose a uWDL handler that can help a service developer to develop context-aware workflow services in ubiquitous computing environments. Figure 2 shows the system structure of the uWDL handler.

After a service developer writes a uWDL workflow service scenario, the uWDL handler compares a context described as subject, verb, and object elements in the uWDL scenario to other contexts, which are obtained as the entity from a sensor network. To do that, we suggest an algorithm to parse uWDL service scenarios and recognize the context according to the sensed contexts from the

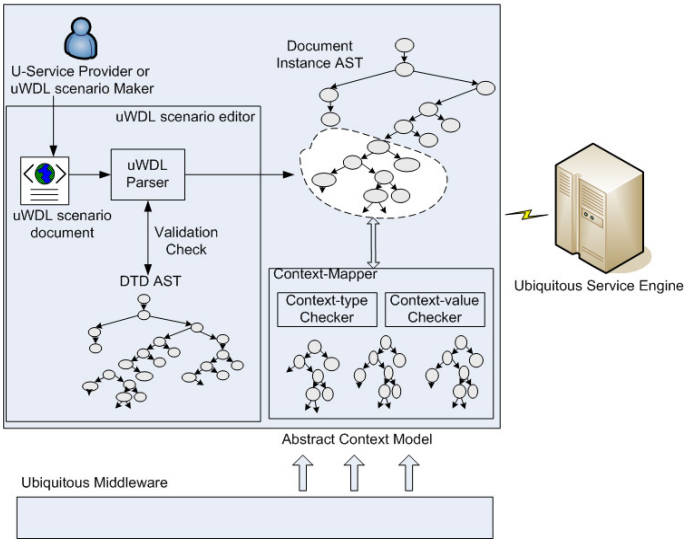

Fig. 2. The architecture for handling the context in uWDL

sensor network. Figure 2 shows a process of manipulating contexts aggregated from the sensor network and providing an adaptive service based on the scenario.

3.2 uWDL Parser

In Figure 2, the uWDL parser analyzes a uWDL service scenario document. A uWDL scenario document is divided into units of token by a lexical analyzer. The tokens are parsed by the uWDL parser, a recursive descendant parser [13, 15], using the uWDL's DTD definition. The uWDL parser examines if a uWDL workflow scenario document is valid to the uWDL DTD AST(Abstract Syntax Tree) [13] that represents a syntax architecture of the uWDL DTD. As a result of a parsing, the uWDL parser makes a DIAST (Document Instance Abstract Syntax Tree) that represents the structure information of a uWDL document as a tree data structure. At this time, a context described as a triple entity in the parsed scenario document is constructed as a subtree in the DIAST produced by the uWDL parser. Figure 3 shows a part of an example DIAST that the uWDL parser creates after it parses a uWDL workflow service document.

In the DIAST tree of Figure 3, a part representing transition conditions that can decide workflow service transitions is a <constraint> area that is marked by a dotted line. That is, a subtree [14, 15] whose root node in the DIAST is a <constraint> element displays that contexts in a workflow service document are

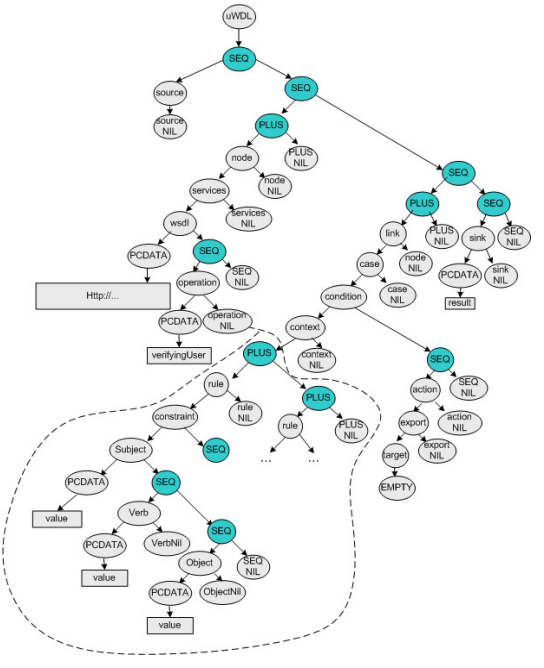

Fig. 3. A part of an example DIAST produced by a uWDL parser

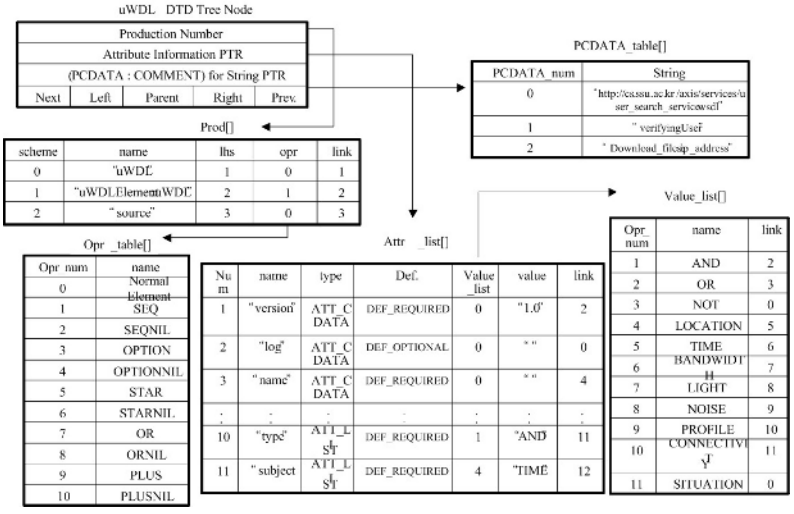

Fig. 4. DIAST¨s element node structure

represented as RDF-based triple nodes. Therefore, a <constraint>'s subtree in the DIAST is compared with a user's situation information inputted from sensors to derive service transitions described in a workflow scenario. Figure 4 displays the data structure of the DIAST's element node and various data tables. The DIAST' nodes are connected to each other and useful data tables by 6 pointers.

Through the data structure of the element node and its data tables, developers can easily know the whole DIAST structure and use it to compare contexts between DIAST and sensors in ubiquitous computing environments. The Production Number is a unique element number which distinguishes each element node. The Left, Parent, and Right links express for element's order and connection information in the DIAST. Each node in the DIAST is divided into a common element node or an operator node. An operator node displays a metacharacter to express a language-specific characteristic of elements in a uWDL's DTD. For example, a parent element node for <node> element in the example DIAST of Figure 3 is <PLUS> operator node, not a common element node. The Attribute Information PTR is a pointer that indicates a relevant record of Attr_list [] to get an element's attribute value. The String PTR is a pointer that indicates a record of the PCDATA_table that contains string information of PCDATA or COMMENT element.

3.3 uWDL Context Mapper and Context Comparison Algorithm

Contexts that the context mapper uses for the comparison are described in a triple entity based on RDF. Context information from the sensor network can be embodied as a triple entity consisting of subject, verb and object according to the structural context model based in RDF. A context described in the

```
Boolean MatchContext(UC A, OCS B) {
  int j; /* For the index of context in B each context set */
  for each j in OCS B { /* Repeatedly comparing contexts in A, B context set */
      if ((A.UCs_type == Bj.OCs_type && A.UCs_value == Bj.OCs_value) &&
          (A.UCv_type == Bj.OCv_type && A.UCv_value == Bj.OCv_value) &&
          (A.UCo_type == Bj.OCo_type && A.UCo_value == Bj.OCo_value))
                return TRUE /* Found context match */
  } /* End for */
  return FALSE; /* Return matchresult */
}
```

Fig. 5. An algorithm for comparing UC A with OCS B

<constraint> element in the uWDL service scenario consists of the triple entity based in RDF. The context mapper extracts context types and values of the entity objectified from sensors. It then compares the context types and values of the objectified entity with those of the DIAST's subtree elements related to the entity. In the comparison, if the context types and values in the entity coincide with the counterpart in the DIAST's subtree, the context mapper drives the service workflow. A context comparison algorithm is shown in Figure 5.

In Figure 5, we define a context embodied with a structural context model from the sensor network as OC = (OCs_type, OCs_value), (OCv_type, OCv_value), (OCo_type, OCo_value), and a context described in a uWDL scenario as UC = (UCs_type, UCs_value) (UCv_type, UCv_value), (UCo_type, UCo_value). OC means a context objectified with the structural context model, and it consists of OCs, OCv, and OCo which mean subject, verb, and object entities. UC means a context described in a uWDL. UCs, UCv, and UCo mean subject, verb, object entities in the uWDL scenario. A context consists of a pair of type and value. Also, OCS and UCS that mean each set of OC and UC can be defined as OCS = (OC1, OC2, OC3, ·, OCi) and UCS = (UC1, UC2, UC3, ·, UCi).

4 Experiments and Results

For testing, we will make a uWDL scenario for an office meeting service in ubiquitous environments, and show how the suggested uWDL handler makes the workflow's service perform context-aware transitions, by comparing contexts described in the scenario with a user's situation information from sensors. The example scenario is as follows. John has a plan to do a presentation in Room 313 at 10:00 AM. When John moves to Room 313 to participate in the meeting before 10:00 AM, a RFID sensor above room 313's door transmits John's basic context information (such as name, notebook's IP address) to a server. If the conditions, such as user location, situation, and current time, are satisfied with contexts described in the uWDL workflow service scenario, then the server downloads his presentation file and executes a presentation program. A developer can use <subject>, <verb> and <object> in the uWDL scenario to decide what

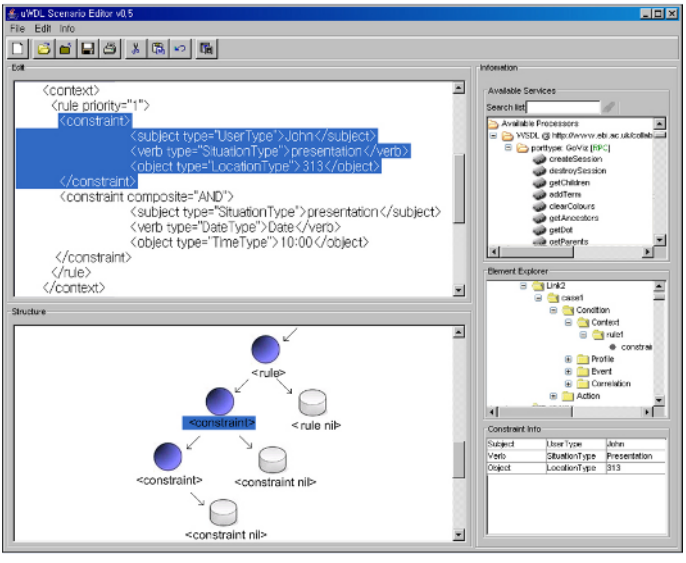

Fig. 6. The service scenario and the DIAST's subtree that the uWDL parser in an uWDL scenario editor produced for the scenario

service is selected, according to context and profile that are the user's situation information.

For experiments, we implement a uWDL scenario editor. The editor includes the suggested uWDL handler and gives convenient user interfaces to set constraints' values of a context. Figure 6 shows a uWDL workflow service scenario for the example scenario, and a part of the <constraint> subtree of DIAST that the WDL parser produces for the uWDL scenario. The subtree in the structure window is for the <constraint> highlighted in the edit window. Now, if the context mapper 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 shown in Figure 6. In this case, because context (UserType, Michael) is not suitable anywhere in the subtree's elements, it is removed. The context described to the uWDL scenario in Figure 6 consists of a limited number of UCs. However, contexts from a sensor network can be produced as innumerable OCs according to the user's situation. Therefore, the uWDL handler must quickly and correctly select an OC coinciding with a UC that is described in the uWDL scenario from such innumerable OCs. In an experiment, we generated a lot of OCs incrementally, and measured how fast the suggested uWDL handler found the OCs that coincided with the UCs in the uWDL scenario of Figure 6. We used a Pentium 4 2.0 Ghz computer with 512M memory based in Windows XP OS for the experiment. Figure 7 is the result.

In Figure 7, we increased the OC's amounts by 50, 100, 200, 300, 400 and 500 incrementally. We placed the OCs coinciding with the UCs in the middle and end of the OCs that we produced randomly. In Figure 7, 1/2 hit-position means

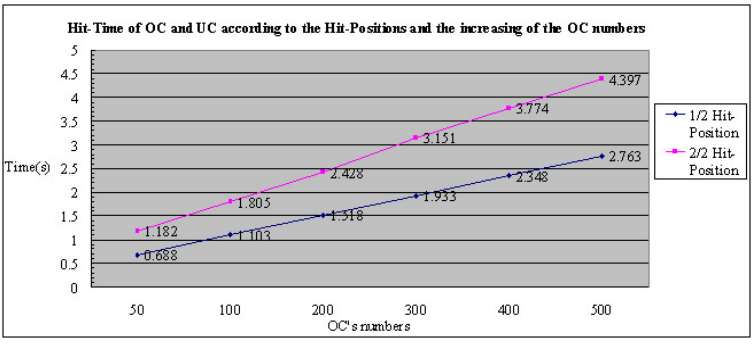

Fig. 7. A hit-time for hit-position and the number of OCs

the position of the OC coinciding with the UC is the middle of the produced OCs, and 2/2 hit-position means the position of the OC is the end of the OCs. As shown in the result, the hit-time did not increase greatly regardless of the OCS's considerable increase. Also, we verified that all schedule.ppt files had been downloaded in all cases when hits between OCs with UCs happened. It shows that the suggested uWDL handler can sufficiently support context-aware workflow service.

5 Conclusion

uWDL is a ubiquitous workflow description language to specify service flows, where appropriate services are selected based on context information and executed concurrently or repeatedly, and to specify context-aware state transition. In this paper, we present the uWDL handler that can process a uWDL workflow service scenario document, and can derive service transition according to a user's situation information. Through experiments, we showed processes in which the uWDL handler parsed the created uWDL scenario document and produced a DIAST for the document. We defined a user's status information from the sensor network as OC based on RDF, and a context described in uWDL scenario as UC. We showed an experiment in which the uWDL mapper compared contexts of UCSs and OCSs through a context comparison algorithm, and measured hit-times and service transition accuracy to verify the efficiency of the algorithm. Through the results, we found that the hit-times were reasonable in spite of the OCs' amounts. Therefore, this uWDL handler will contribute greatly to the development of the context-aware application programs in ubiquitous computing environments.

Acknowledgements

This research is supported by the Ubiquitous Autonomic Computing and Network Project, the Ministry of Information and Communication (MIC) 21st Century Frontier R&D Program in Korea.

References

1. 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)
2. M. Weiser: Some Computer Science Issues in Ubiquitous Computing. Communications of the ACM, Vol.36, No.7 (1993) 75-84
3. D. Hollingsworth: The Workflow Reference Model. Technical Report TC00-1003, Work flow Management Coalition (1994)
4. Guanling Chen, David Kotz: A Survey of Context-Aware Mobile Computing Research, Technical Report, TR200381, Dartmouth College (2000)
5. Tony Andrews, Francisco Curbera, Yaron Goland: 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. R. Scott Cost, Tim Finin: ITtalks: A Case Study in the Semantic Web and DAML+OIL. University of Maryland, Baltimore County, IEEE (2002) 1094-7167
9. W3C: RDF/XML Syntax Specification, W3C Recommendation (2004)
10. James Snell: Implementing web services with the WSTK 3.2, Part 1, IBM Tutorials, IBM (2002)
11. Jena2-A Semantic Web Framework.
    Available at http://www.hpl.hp.com/semweb/jena1.html
12. Deborah L. McGuinness, Frank van Harmelen (eds.): OWL Web Ontology Language Overview, W3C Recommendation (2004)
13. Aho, A.V., Sethi R., and Ullman J. D., Compilers: Principles, Techniques and Tools, Addison-Wesley (1986)
14. Bates, J. and Lavie A.,"Recognizing Substring of LR(K) Languages in Linear Time", ACM TOPLAS, Vol.16 ,No.3, pp.1051-1077 (1994)
15. Reckers J. and Koorn W., Substring parsing for arbitrary context-free grammars. ACM SIGPLAN Notices,, 26(5), pp.59-66 (1991)