WS-CDL 개발 동향 및 모바일 환경을 위한 개선 방향

이강찬* · 이원석* · 전종홍* · 이승윤* · 박종헌**

*한국전자통신연구원, **서울대학교

The Trends of Web Services Choreography and its enhance model for Mobile Environments

Kangchan Lee* · Wonsuk Lee* · Jonghong Jeon* · Seungyun Lee* · Jonghun Park**

*Eletronics and Telecommunications Research Institute, **Seoul National Universoty

E-mail : {chan, wslee, hollobit, syl}@etri.re.kr* · jonghun@snu.ac.kr

ABSTRACT

하나의 웹서비스가 이와 연동하는 클라이언트와 1회 이상의 연동(interaction)을 필요로 할 경우, 웹서비스에서는 이를 명확히 정의하는 규약이 필요하게 되는데 이를 안무(choreography) 또는 대화(conversation)라 부르며, 총칭하여 협업 웹서비스 기술로 부른다. 최근 W3C는 웹서비스의 안무를 위한 WS-CDL(Choreography Description Language)를 개발 중에 있으며, WS-CDL은 웹서비스들 간의 전체적인 안무 교환이나 메시지 교환을 정의 한다. 그러나, WS-CDL은 연결성(connectivity)이 항상 보장되지 않는 모바일 환경이나 사용자가 이동하는 경우에 사용하기에는 비효율이 따른다. 본 논문에서는 WS-CDL 개발 동향에 대해서 살펴 보고, 가지고 있는 문제점을 보완하기 위한 방안을 제시한다.

키워드

Web Services, Choreography, Preferences, Mobile Web Services

I. Introduction

WS-CDL (Web Services Choreography Description Language) version 1.0 from W3C is an XML-based language that describes peer-to-peer collaborations of web services participants through defining, from a global viewpoint, their common and complementary observable behavior, where ordered message exchanges result in accomplishing a common goal.

While web service is currently emerging as a dominant technology for supporting e-Business automation and integration, it is also increasingly considered as a promising platform for inter-connecting devices in mobile and ubiquitous computing environment. By embedding the web services into virtually any computing devices, it becomes possible for a device to discover and interoperate with other devices and external services, establishing pervasive network of computers of all form factors and wireless devices. Indeed, considering that the interoperability problem is the crux for realizing the vision of ubiquitous computing and the web services are meant to be consumed by programs not humans, we envision that making every device an autonomous web service appears to be a vital approach. Some of the current ongoing efforts along this line include Microsoft's invisible computing project, UPnP 2.0, and OMA's mobile web services working group.

In particular, when a mobile device is web service enabled and engages in a conversation with a service provider, it becomes necessary to define a interaction logic required for the client. For this purpose, a choreography language can be used to provide the rules of engagement between the mobile client and the web service provider. 3

In this mobile service environment, however, connection may be lost or the mobile device may move into out-of-service area any time during the conversation, and this may prevent

the conversation from successful completion particularly when the conversation is long-running or involves user interactions. Accordingly, performing mere step by step execution of a choreography specification defined for the mobile client may yield unsatisfactory performance results.

To address these problems, we propose a flexible framework, named Web Services Conversation Preference Profile (WS-CPP), through which mobile web service clients can express their preferences on how conversation should take place while being able to satisfy a choreography specification. The proposed framework allows some of the activities defined in a choreography representation to be selectively skipped, providing an effective means to flexibly reduce the number of messages exchanged between the mobile client and web service provider.

II. Preferences on WS-CDL Specification

The proposed preference specification model, WS-CPP, bases the WS-CDL as a choreography language, and focuses on the specific preference requirements that can arise during the multi-step interactions with a web service provider. It enables web service clients to express their interaction preferences in a standard format that can be delivered to and interpreted by service providers.

Given a WS-CDL description that represents a set of valid interaction sequences, WS-CPP allows conversation preferences to be associated with some of the interactions defined in the WS-CDL so that they are not required during actual conversations. Specifically, from the WS-CDL entities, we identify a set of activities that can be associated with preferences as follows.

An activity notation in WS-CDL is the lowest level component of a choreography, and it is used to define an activity as either an ordering structure, a work unit, or a basic activity. An ordering structure consists of sequence, parallel, and choice, and it combines activities with other ordering structures in a nested way to specify the ordering rules of activities.

All activities enclosed within the sequence need to be executed sequentially in the same order that they are defined, and are not allowed to be skipped. In contrast, the parallel structure contains one or more activity notations that are enabled concurrently, and a preference can be introduced to specify the execution priorities among the activity notations within the parallel structure. Similarly, when two or more activity notations are specified within a choice element, requiring only one of them to be performed, a preference on which activity notation is to be selected can be introduced.

A workunit is used to describe a guard condition as well as a repetition condition on an activity notation. The guard condition expresses the interest on one or more variable information to be available under certain prescribed constraints. A workunit completes successfully when its repetition condition evaluates to be false. Since the number of iterations executed for a workunit can be controlled by the messages sent from a client in some cases, we define a preference notation that allows a client to specify a set of values that are pre-assigned to the variables of interest for the purpose of minimizing the number of interactions resulting from a loop execution.

As for the basic activity which contains interaction, perform, assign, silent action, no action, and finalize activities, we identify two basic activities that can be associated with preferences. The interaction activity is used to exchange information between collaborating parties, and in particular the message exchange is specified in exchange element within the interaction. Therefore, a preference indicating whether a specific message exchange should take place or not can be defined for the exchange element.

On the other hand, for the assign activity that is used to create or change the value of one or more variables in a WS-CDL document, we define a preference that allows the value of a variable to be assigned beforehand in order to make some of the interactions defined in the choreography unnecessary.

Having discussed the conversation preferences defined in WS-CPP, we now proceed to describe a required run-time behavior when a WS-CPP profile is to be used. First, we assume that a WS-CDL document is publicly available from a web service provider so that it can be referred to by a client application's developer. The developer then needs to write a client application of which the interaction behavior conforms to the requirements specified in the service provider's WS-CDL. The client program also must be WS-CPP aware. In other words, it must be able to interpret the WS-CPP and adjust its behavior according to the WS-CPP.

Later, a WS-CPP document that reflects the client application's preferred conversation behavior can be defined by a user agent, and then it can be transmitted to a service provider when an actual conversation starts.

In the meantime, after the service provider has received a WS-CPP profile, it should be able to refer to the preference specifications in the WS-CPP throughout the conversation. That is, while the service provider engages in a conversation according to the prescribed WS-CDL, it needs to look up the preference definition from the WS-CPP document for each activity notation that can be associated with a preference so that it can seamlessly interact with the client without raising exceptions.

The resulting behavior will be that the number of messages exchanged between the client and the service provider under the proposed WS-CPP framework will be always equal to or less than that of the original WS-CDL definition as illustrated in Figure 1.

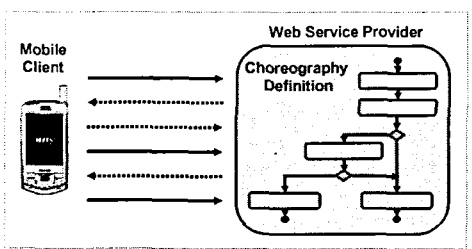

Figure 1. Conversation behavior under WS-CPP

III. WS-CDL Structure

A WS-CPP document consists of preference definitions. We have adopted XPath (www.w3.org/TR/xpath) to express conditions as well as to locate the elements in the WS-CDL and WS-CPP documents. The top level element of a WS-CPP document is preference that has author information, namespace definitions, and a reference to the target WS-CDL document. The syntax of the preference construct is:

```
<preference
  name="ncname"
  author="xsd:string"?
  version="xsd:string"?
  root = "true" | "false"?
  choreographyDefinition="XPath expression to a choreography tag"
  targetNamespace="uri"
  xmlns=http://localhost/ws-chor/cpp>
  Preference-Notation*
</preference>
```

The Preference-Notation can be one of the following elements: variableDefinitions, interactionSkip, workunitSkip, choicePriority, and orderPriority. The variableDefinitions construct allows defining one or more variables to be used within a preference document. A variable defined in WS-CPP points to a data structure within informationType in WS-CDL by using an XPath expression so that its value can be referred to for instantiating a corresponding variable in WS-CDL. The syntax of the variableDefinitions is:

```
<variableDefinitions>
  <variable name = "ncname" type= "qname" query = "XPath expression" />+
</variableDefinitions>
```

The interactionSkip of WS-CPP represents a preference on the exchange element within an interaction activity in the WS-CDL. The target exchange element is referred to by use of an XPath expression pointing to the element in the WS-CDL document. It indicates that a message expected to be delivered to a receiver will not be actually sent. Instead, the receiver should presume as if it were received. This is accomplished by providing all data required from the message with preAssignment element that pre-assigns a value to a variable defined in a WS-CDL document so that the actual interaction becomes unnecessary at runtime. In order to distinguish the client-initiated exchange from the service provider-initiated exchange, we use an attribute named type. The syntax of the interactionSkip construct is:

```
<interactionSkip name = "ncname"
  guard = "xsd:boolean XPath expression"?
  target = "XPath expression to an exchange tag"
  type = "ignore" | "filter">
  <preAssignment name = "ncname">
    <copy source = "XPath-expression" target = "qname" />+
  </preAssignment>?
</interactionSkip>
```

The above preference definition employs a guard to express a condition that must be satisfied in order for the preference to be activated. The type with the value ignore indicates that the interaction to be skipped is a client-initiated exchange, and the type with the value filter represents the other case.

The workunitSkip is used when a client wants to

avoid iterative interactions required by a workunit activity notation. By use of the workunitSkip, the client may pre-assign multiple values to a variable defined in a WS-CDL document for the purpose of avoiding sending messages repetitively to exit the loop. In case that the loop is not exited after trying a list of possible values, the interactions in the workunitSkip may proceed as defined in the WS-CDL or the entire conversation may be stopped depending on the value of causeStop. The syntax of workunitSkip is shown below:

```
<workunitSkip name = "ncname"
  guard = "xsd:boolean XPath expression"?
  target = "XPath-expression to an exchange tag in workunit"
  causeStop = "true" | "false">
  <multiPreAssignment name = "ncname">
      <copy source = "list of XPath-expressions" target = "qname" />+
  </multiPreAssignment>
</workunitSkip>
```

The choicePriority allows the selection to be pre-specified when the client is required to make a decision among the available choices defined in the WS-CDL specification. A priority can be defined in terms of a string that refers to the position of an activity element within a choice structure. The syntax of choicePriority element is:

```
<choicePriority name = "ncname"
  guard = "xsd:boolean XPath expression"?
  target = "XPath-expression to a choice tag" >
  <selection position = "xsd:string" />
</choicePriority>
```

Finally, the orderPriority corresponding to the parallel structure specifies the client's execution ordering priority among the activities that can be enabled in parallel. The priorities among the activities are specified in terms of a series of partial order expressions that must hold for activity pairs.10 The syntax is defined as follows:

```
<orderPriority name = "ncname"
  guard = "xsd:boolean XPath expression"?
  target = "XPath-expression to a parallel tag" >
  <priority expression = "Prioritization expression" />+
</orderPriority>
```

IV. Conclusion

A conversation preference specification framework for WS-CDL, called WS-CPP, was proposed to enhance the performance of mobile web services applications as well as to support flexibility in web services conversation. WS-CPP allows some of the interactions defined in a WS-CDL document to be skipped while satisfying the rules of conversation required by service providers.

In addition, it provides a means for the mobile clients to pre-specify their preferences on the choices and the concurrencies that can arise during a conversation with service providers. We are currently working on to extend the current work so that it can support additional WS-CDL features such as loops and exceptions. Further work is also required to specify a protocol for delivering WS-CPP specifications and to specify how a profile should be processed by a WS-CDL processor.

References

[1] N. Kavantzas, D. Burdett, G. Ritzinger, T. Fletcher, and Y. Lafon. (2004) Web services choreography description language version 1.0. [Online]. Available: http://www.w3.org/TR/2004/WD-ws-cdl-10-20041217/

[2] Microsoft Corp. (2004) XML web services on a chip. [Online]. Available :http://research.microsoft.com/invisible/default.asp

[3] UPnP Forum. (2005) UPnP 2.0. [Online]. Available:http://www.upnp.org/

[4] T. Goddard. (2005) Using the network configuration protocol (NETCONF) over the simple object access protocol (SOAP). [Online]. Available: http://www.ops.ietf.org/netconf/

[5] G. Klyne, F. Reynolds, C. Woodrow, H. Ohto, J. Hjelm, M. H. Butler, and L. Tran. (2004) Composite capability/preference profiles (CC/PP): Structure and vocabularies 1.0. [Online]. Available: http://www.w3.org/TR/2004/REC-CCPP-structvocab-20040115/

[6] C. Peltz, "Web services orchestration and choreography", IEEE Computer, Volume: 36, Issue: 10, Oct. 2003, pp. 46 - 52.