WS-CPP 프리퍼런스 모델

이강찬* · 이원석* · 전종홍* · 이승윤* · 박종헌**
*한국전자통신연구원, **서울대학교

WS-CPP(Web Services Conversation Preference Profile) Preference Model

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

The Web Services Choreography Description Language (WS-CDL) is an XML-based language that describes peer-to-peer collaborations of parties by defining, from a global viewpoint, their common and complementary observable behavior; where ordered message exchanges result in accomplishing a common business goal.
In this paper, we survey and analysis the functionality of the WS-CDL, and propose new language, which enhance the WS-CDL for the conversation the message between entities

키워드

Web Services, Choreography, Preferences, Mobile Web Services

Ⅰ. Introduction

A choreography description may be used to generate the necessary code skeletons that can be said to implement the required external observable behavior for that Web Service. A choreography description may also be used to aid the testing of participating Web Services through the generation of test messages that could be sent to participants by means of an appropriate vendor specific tool that reads the choreography description and manages the test interaction according to the choreography description. A choreography description may also be used to validate the multi-party observable interactions amongst a collection of Web Services. A choreography description may also be used to show the presence of useful properties such as lock freedom and leak freedom in the behavioral contract. In this sense a choreography description acts as a model of the behavior across a number of Web Services which in turn can be subject to static analysis to show that if and only if the underlying Web Services behave according to the contract that the interaction between the Web Services will exhibit these properties.

A WS-CDL document can be used at design time by a participant to verify that its internal processes will enable it to participate appropriately in the choreography. It can also be used to develop a web services-based composite process that can be said to implement the required external observable behavior for the process. At run time, the choreography definition can be used to verify that everything is processed according to the predefined conversation protocol [1].

Some of the current ongoing efforts along this line include Microsoft's invisible computing project [2], UPnP 2.0 [3], and NETCONF [4]. When a mobile device is web services enabled and engages in a conversation with a service provider, it becomes necessary to define a choreography for the collaborating parties. For this purpose, the WS-CDL can be used to provide the rules of engagement between the mobile client and the web service provider. In this mobile services 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 produce unsatisfactory performance results.

II. Preference Model for WS-CPP

This section introduces the proposed preference specification model for WS-CDL. WS-CPP 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 notation, 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 in order 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 in a choice element, requiring only one of them to be performed, a preference on which activity notation is to be selected can be introduced.

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 be carried out 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 writes a client application of which the interaction behavior conforms to the requirements specified in the service provider's WS-CDL. Subsequently, 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 WSCPP profile, it will 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 pre-defined 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 issuing any 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.

III. WS-CDL Structure

A WS-CPP document consists of a set of

definitions. The top level element of a WS-CPP document is preference that governs the interaction behavior of the service provider and client. A preference may contain zero or more of the following entities: interactionSkip, choicePriority, and orderPriority. 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 WSCDL 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 achieved by providing all data necessary from the message with preAssignment element which pre-assigns a value to a variable defined in a WS-CDL document so that the actual interaction becomes not necessary. In order to distinguish the client-initiated exchange from the service provider initiated exchange, we use the attribute type. The syntax of the interactionSkip construct is:

[redacted]

The `choicePriority` allows the selection to be prespecified when the client is required to make a decision among the available choices defined in a WS-CDL specification. A priority can be defined in terms of either a specific order within choice activity or an XPath expression that refers to an activity element. The syntax of `choicePriority element` is:

[redacted]

Finally, `orderPriority` in WS-CPP corresponding to the parallel of the WS-CDL 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 the order in which the priority element appears within the `orderPriority`. The syntax is defined as follows:

IV. WS-CPP Example

In this section, we consider the following simple scenario to demonstrate a usage of WS-CPP: A mobile device may be temporarily within the range of wireless LAN which provides two types of on-demand multimedia streaming services, namely a regular service and a premium service. In order to start the service, the device first needs to invoke the service, and then it is required to deliver some context information such as the screen size and the media handling capability of the device to the service provider. For simplicity, we assume that only the screen size information is required. When the service provider receives the configuration data, it immediately replies back to the client with ACK message, and then it checks if the configuration is valid. In case that the configuration is not valid, the service provider notifies the client and the choreography complete. Otherwise, the device may choose the service type, and subsequently the choreography completes after the service provider sends ACK message. The example scenario is illustrated in Figure 1, and the corresponding service provider's WS-CDL document is sketched in Figure 2.

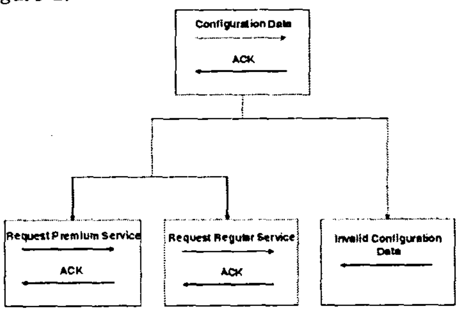

Figure 1: Example scenario

- *P*1 The client wants to skip the interaction for sending configuration data by providing them in the WS-CPP document.
- *P*2 The client prefers not to receive an ACK message for the configuration data transmission.
- *P*3 The client prefers the premium service to the regular service

The resulting WS-CPP specification is given in Figure 3. In this example WS-CPP, the preference *P*1 is represented by `interactionSkip` where the target WS-CDL element that needs to be skipped is defined by use of an XPath expression and the necessary configuration data is pre-defined in the `preAssignment` element. Since the WS-CPP document is delivered to the service provider when the client invokes the service, the actual interaction in a conversation becomes unnecessary.

```
<choreograph
 • • •
  <sequence>
   <interac
    <exchai
```

Figure 2: WS-CDL definition for the example

The second `interactionSkip` element of the WS-CPP profile expresses the preference *P*2. The type `filter` is used for this case as the message is supposed to be sent by the service provider. Similarly, the preference *P*3 is specified by indicating that the first `interaction` within the `choice` element of the WS-CDL document needs to be selected. Hence, it is clear from this example that the WS-CPP provides an effective means to flexibly reduce the number of messages exchanged between the mobile clients and service providers.

```
<package nam
 • • •
  <preferenc
   root="tr
```

Figure 3: WS-CPP definition for the example

V. 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-struct vocab-20040115/

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