Incorporating Preferences into Web Service Conversations

Jonghun Park
Seoul National University
Seoul, Korea
jonghun@snu.ac.kr

Wan Lee
Seoul National University
Seoul, Korea
wanlee80@gmail.com

Kangchan Lee
Elec. & Telecom. Res. Inst.
Daejeon, Korea
chan@etri.re.kr

ABSTRACT

Recently web services choreography working group of W3C has published the working draft on WS-CDL (Web Services Choreography Description Language) version 1.0 which defines reusable common rules to govern the ordering of exchanged messages between web services participants. This paper considers a computing environment where mobile clients are interacting with web service providers based on a WS-CDL specification. In order to effectively cope with the user and device mobility of such an environment, in this paper we present an ongoing work to develop a framework through which a mobile client can specify its preference on how conversation should take place. The proposed framework provides a flexible means for mobile clients to minimize the number of message exchanges while allowing them to adhere to the required choreography.

Keywords

Web Services, Choreography, Preferences, Mobile Web Services

1. INTRODUCTION

Web services choreography aims at the coordination of interactions between distributed parties, which define web services to expose their externally accessible operations. The coordination of interactions becomes necessary in almost any autonomous web services conversations, including the areas ranging from simple data processing to complex supply chain management. Motivated by the need for an effective mechanism to coordinate the interactions among web services and their user agents, World Wide Web Consortium has formed the web services choreography working group that has been tasked with the development of such a mechanism in an interoperable way.

As a result, the working group has recently published the working draft of web services choreography description language (WS-CDL) version 1.0 that consists of web services choreography requirements, web services choreography model overview, and web services choreography description language, aiming to provide the capability of composing interoperable collaborations between any type of party regardless of the supporting platform or programming model. The WS-CDL 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 [1].

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].

While web services is currently emerging as the dominant application on the Internet for facilitating 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 automatically discover and interoperate with other devices, establishing pervasive peer-to-peer network connectivity of computers of all form factors and wireless devices. Indeed, considering that the interoperability problem is the crux of realizing the vision of ubiquitous computing and the web services are meant to be consumed by programs not humans, 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 [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.

Motivated by the above remarks, this paper proposes a flexible framework, named Web Services Conversation Preference Profile (WS-CPP), through which mobile web services clients can express their preferences on how conversation should take place while being able to adhering to a choreography specification. The proposed framework assumes that a choreography is defined in terms of WS-CDL, and allows some of the activities defined in the WS-CDL representation to be selectively skipped, providing an effective means to flexibly minimize the number of messages exchanged between the web service client and provider. Our work presented in this paper can be compared to CC/PP (Composite Capability / Preference Profiles) that describes device capabilities and user preferences to guide the adaptation of content presented to the device [5]. Both of them attempt to specify the client-side preferences on how the client and service provider should interact with each other. However, while the presented scheme focuses on the specific preference requirements that can arise during the multi-step interactions with dynamic web service resources, the preferences addressed in CC/PP only consider the static web resources such as html documents and multimedia files. The paper is organized as follows: Section 2 provides an overview of the proposed preference model and framework architecture. Section 3 describes the structure of a WS-CPP specification, and defines the WS-CPP entities. In Section 4, a simple example is presented to show usage of WS-CPP. Finally, Section 5 concludes the discussion.

2. PREFERENCE MODEL FOR WS-CDL

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 WS-CPP 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.

3. WS-CPP 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 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 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:

```
<interactionSkip name = "ncname"
    guard = "xsd:boolean XPath-expression"?
    target = "XPath expression to an exchange tag"
    type = "ignore" | "filter">
    <preAssignment name = "ncname"
        target = "XPath expression to a variable name"
        <value variable = string | number |
            "XPath expression"/>
    </preAssignment>*
</interactionSkip>
```

`choicePriority` allows the selection to be pre-specified 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:

```
<choicePriority name = "ncname"
    target = "XPath expression to a choice tag"
    selection = "number" | "XPath expression"/>
</choicePriority>
```

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:

```
<orderPriority name = "ncname"
    target = "XPath expression to a parallel tag"
    priority = "number" | "XPath expression"/>+
</choicePriority>
```

4. 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 completes. 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.

In order to show a usage example of WS-CPP, we now make the following assumptions on the client's conversation preferences.

$P1$ The client wants to skip the interaction for sending configuration data by providing them in the WS-CPP document.

$P2$ The client prefers not to receive an ACK message for the configuration data transmission.

$P3$ 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 $P1$ 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.

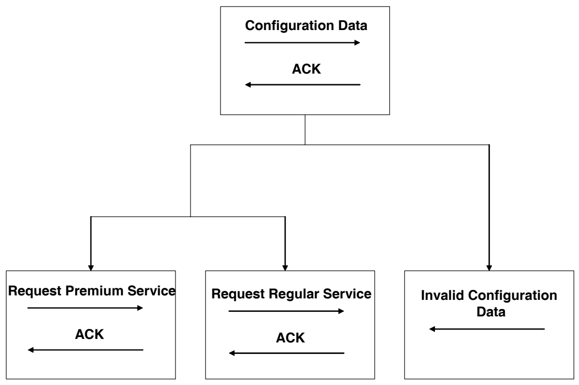

Figure 1: Example scenario

```
<choreography name="MultimediaServiceChoreo">
  ...
  <sequence>
    <interaction>
      <exchange>Send Configuration Data</exchange>
      <exchange>ACK</exchange>
    </interaction>
    <choice>
      <choice>
        <interaction>
          <exchange>Request Premium Service</exchange>
          <exchange>ACK</exchange>
        </interaction>
        <interaction>
          <exchange>Request Regular Service</exchange>
          <exchange>ACK</exchange>
        </interaction>
      </choice>
      <interaction>
        <exchange>Invalid Configuration Data</exchange>
      </interaction>
    </choice>
  </sequence>
</choreography>
```

Figure 2: WS-CDL definition for the example

The second `interactionSkip` element of the WS-CPP profile expresses the preference $P2$. The type `filter` is used for this case as the message is supposed to be sent by the service provider. Similarly, the preference $P3$ 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.

5. CONCLUSION

This paper discussed an overlay preferencing mechanism that ultimately alleviates additional real time messaging requirements during web services based interactions. Specifically, a conversation preference specification framework for WS-CDL, called WS-CPP, was proposed to enhance the per-

```
<package name="MultimediaServiceCPP"
  ...
  <preference name="multimediaServicePref"
    root="true"
    refer="cns:/package/choreography[1]">
    <interactionSkip name="skipConfiguration"
      target="/sequence[1]/interaction[1]/exchange[1]"
      type="ignore">
      <preAssignment name="screenSize"
        target="cns:/screenSize"
        <value variable="240x320x18"/>
      </preAssignment>
    </interactionSkip>
    <interactionSkip name="skipConfigurationACK"
      target="/sequence[1]/interaction[1]/exchange[2]"
      type="filter"/>
    <choicePriority name="preferPremiumService"
      target="/sequence[1]/choice[1]/choice[1]">
      <priority variable="/interaction[1]"/>
    </choicePriority>
  </preference>
</package>
```

Figure 3: WS-CPP definition for the example

formance 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.

6. 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/