Defining Semantic Guideline in XML-based Programmable Service Environment

Pattara Leelaprute[1], Masahide Nakamura[2], Ken'ichi Matsumoto[2] and Tohru Kikuno[1]
[1]*Graduate School of Information Science and Technology, Osaka University, Japan*
{pattara, kikuno}@ist.osaka-u.ac.jp
[2]*Graduate School of Information Science, Nara Institute of Science and Technology, Japan*
{masa-n, matumoto}@is.aist-nara.ac.jp

1. Introduction

Traditionally, network services have been deployed by service providers in a *ready-made* form. The implementation and configuration of the services are carefully performed by expert staffs. The end users just *subscribe to* the ready-made services to use them. For instance, the conventional telephony services, like Call Forwarding and Call Number Blocking, are managed by telecom carriers.

Recently in the Internet, a paradigm shift in managing services occurs with standardized protocols and powerful terminals with intelligent agents. The end users (or third parties) can create and control their own services within the local servers. Thus, the services become more and more *programmable*, which extends the range of the user's choice and flexibility. The programmable service is generally specified with a certain language. In recent years, many XML-based languages are proposed for the service description (e.g., WSDL for Web Services, CPL for VoIP[1], VoiceXML for interactive voice services).

Most of these XML-based languages are *syntactically* defined by DTDs or XML Schemas. However, these do not usually cover the *semantic aspect* of the programmable services. Indeed, the compliance with the DTD or the XML Schema is *not* a sufficient condition for the correctness of the service. Since the end users are not as expert as the conventional service providers, there is enough room for *naive users* to create semantic flaws in the service. This significantly reduces quality and dependability of the services.

To cope with this problem, it is essential to establish a *semantic guideline*, by which the end users can debug the service and validate its semantic correctness. In our previous research[2], we have proposed a notion of *semantic warnings* for Call Processing Language (CPL) of VoIP. The idea was borrowed from the program compilation, which identifies problematic structures in the code (XML text in this context). The warnings are not necessarily errors, but often lead the service to undesirable behaviors as shown in our experiment. In the research, we have defined eight classes of warnings. However, there is no guarantee that these eight classes are complete. Further classes of warnings may be found in the future. Therefore, for the XML-based programmable service environment (not limited to the CPL), we consider it inevitable to have a more general framework, by which we can specify the semantic guidelines themselves in a programmable way.

For this purpose, this paper presents an application of an XPath-based language, SGSL (Simple Guideline Specification Language) [3], which was originally developed to formulate the Web Contents Accessibility Guideline by W3C. Specifically, by describing the CPL semantic warnings in the SGSL, we examine what features are essentially needed for the general semantic guideline.

2. CPL semantic warnings

The CPL allows a user to describe freely how his/her calls should be processed by the VoIP server. The user can specify the signaling action (proxy, redirect, or reject), as well as the next location where the call is to be directed. The user can also specify conditional branches using switches based on address, time, etc (see [1]). Focusing on the semantic aspects of telephony services, we defined eight types of semantic warnings in [2]. Due to limited pages, we here present only three of the eight warnings.

Call rejection in all execution paths (CRAE)
Definition: All execution paths terminate at `<reject>`.
Example: By the script in Figure 1(a), any incoming call is rejected, no matter who the originator is. All actions and evaluated conditions are in vain after all, which just wastes resources of the VoIP servers.

Address set after address switch (ASAS)
Definition: When `<address>` and `<otherwise>` tags are specified as outputs of `<address-switch>`, the same address evaluated in the `<address>` is set in the `<otherwise>` block.
Example: When the user makes an outgoing call, the script in Figure 1(b) checks the destination of the call. The call is rejected if the destination is `bob@example.com`. How-

```
<?xml version="1.0" ?>
<!DOCTYPE cpl SYSTEM "cpl.dtd">
<cpl>
  <incoming>
    <address-switch field="origin" subfield="user">
      <address is="anonymous">
        <reject status="reject" reason=
         "I don't accept anonymous calls"/>
      </address>
      <address is="guest">
        <reject status="reject" reason=
         "I don't accept call from guest"/>
      </address>
      <otherwise>
        <reject status="reject" reason=
         "I don't accept call from anyone"/>
      </otherwise>
    </address-switch>
  </incoming>
</cpl>
```

(a) Example of CRAE

```
<?xml version="1.0" ?>
<!DOCTYPE cpl SYSTEM "cpl.dtd">
<cpl>
  <outgoing>
    <address-switch field="original-destination"
     subfield="user">
      <address is="sip:bob@example.com">
        <reject status="reject"
         reason="I don't want to call Bob" />
      </address>
      <otherwise>
        <location url="sip:bob@example.com">
          <proxy/>
        </location>
      </otherwise>
    </address-switch>
  </outgoing>
</cpl>
```

(b) Example of ASAS

```
<?xml version="1.0" ?>
<!DOCTYPE cpl SYSTEM "cpl.dtd">
<cpl>
  <incoming>
    <address-switch field="origin" subfield="user">
      <address subdomain-of="family">
        <location url="sip:mobile@example.com">
          <proxy/>
        </location>
      </address>
      <address subdomain-of="friend">
        <location url="sip:mobile@example.com">
          <proxy/>
        </location>
      </address>
      <otherwise>
        <location url="sip:mobile@example.com">
          <proxy/>
        </location>
      </otherwise>
    </address-switch>
  </incoming>
</cpl>
```

(c) Example of IASS

Figure 1. Example CPL scripts

ever, for other destination address, the call is proxied to `bob@example.com`, which must have been rejected.

Identical actions in single switch (IASS)

Definition: The same actions are specified for all conditions of a switch.

Example: The script in Figure 1(c) specifies a conditional branch of the caller's address (origin). However, whatever the address is, the call is proxied to `mobile@example.com`. Hence, the switch is redundant.

Note that neither DTD nor XML Schema of the CPL can identify the semantic warnings. Indeed, all scripts in Figure 1 are syntactically valid against them.

3. Defining semantic warnings with SGSL

The semantic guidelines (including our warnings) are often defined on the XML's tree structure. Specifically, they are characterized by properties on a certain set of nodes in the XML text. For this, the XPath could be an appropriate means, since it can specify Boolean expressions (XPath-BooleanExpresion) for the node properties, and expressions for addressing any set of nodes (XPathNodeSetExpression). However, the XPath basically evaluates the XML text in a *path basis*. So, it cannot describe semantic guidelines defined over the tree structure (e.g., CRAE).

Therefore, we attempt to apply a language called SGSL (Simple Guideline Specification Language)[3]. The SGSL extends the XPath so that it can manipulate multiple nodes in different paths using `some` and `every` operators (see Figure 2). It was originally designed to specify the Web Content Accessibility Guidelines by W3C, such as "Provide the ALT attribute for every IMG elements". Using the SGSL, the semantic warnings CRAE and ASAS in Section 2 can be described as follows:

CRAE: `if every $x in [count(node())=0] satisfies $x=reject then warn("CRAE")`

Description: if every terminating node (having 0 children) is equal to `reject`, then issue the warning CRAE.

ASAS: `if some $x in //address-switch satisfies $x/[address/@is = otherwise/location/@url] then warn("ASAS")`

Description: For some child node of `address-switch`, if the attribute `is` in `address` is equal to the attribute `url` in `location` in `otherwise`, then issue the warning ASAS.

We found it impossible to specify IASS with the SGSL, since neither the XPath nor the SGSL is capable of comparing *sub-trees* in the XML text. After all, we confirmed that all warnings except IASS can be specified by the SGSL.

In this research, it was shown that the XPath and even its extension, the SGSL, could not achieve complete coverage of the CPL semantic warnings. This implies that a more expressive language is necessary to specify semantic guidelines for general XML-based programmable services. Through the application to the CPL, it was seen that what lacked for the SGSL was the *tree-wise comparison*, where the expression can test equality and inclusion of sub-trees in the XML text.

Of course, the semantic guidelines vary depending on the target services, so other necessary features still might exist for different services. However, we believe that there must exist common semantic properties for general XML-based programmable services. Extracting such properties and generalizing them for a universal semantic guideline framework (e.g., language) are a quite crucial problem, which is our research goal.

```
Checkpoint::= BooleanExpression
            | if BooleanExpression then warn(String)
BooleanExpression::= XPathBooleanExpression
  | every Variable in NodeSet satisfies BooleanExpression
  | some Variable in NodeSet satisfies BooleanExpression
NodeSet::= XPathNodeSetExpression
```

Figure 2. Syntax of the SGSL (in BNF)

References

[1] J. Lennox and H. Schulzrinne, "CPL:A Language for User Control of Internet Telephony Service", Internet Engineering Task Force, Jan 2002, http://www.ietf.org/internet-drafts/draft-ietf-iptel-cpl-06.txt

[2] M. Nakamura, P. Leelaprute, K. Matsumoto and T. Kikuno, "On Detecting Feature Interactions in Programmable Service Environment of Internet Telephony", *Journal of Computer Networks*, Elsevier, (to appear).

[3] Y. Takata, T. Nakamura and H. Seki, "Automatic Accessibility Guideline Validation of XML Documents Based on a Specification Language", *10th Int'l Conf. on Human-Computer Interaction (HCI 2003)*, Vol.4, pp.1040-1044, June 2003.