XML BASED FRAMEWORK FOR CONTACT CENTER APPLICATIONS

Nikolay Anisimov, Brian Galvin and Herbert Ristock

Genesys Telecommunication Laboratories (an Alcatel-Lucent company)
2001 Junipero Serra, Daly City, CA, USA

Keywords: Call center, contact center application, VoiceXML, Call Control XML, organizational structure.

Abstract: W3C languages, VoiceXML and CCXML all play an important role in contact centers (CC) by simplifying the creation of CC applications. However, they cover only a subset of contact center functions, such as simple call control and interactive voice response (IVR) with automatic speech recognition. We discuss ways to complement VoiceXML and CCXML in order to cover all necessary contact center functions required to script end-to-end interactions in a consistent and seamless way. For this purpose we introduce an XML forms-based framework called XContact comprising the CC platform and applications, multi-script and multi-browsing, and interaction data processing. We also discuss how routing as a key CC capability can be scripted/captured within such framework in order to demonstrate the overall approach.

1   INTRODUCTION

Contact centers (CC) play a very important role in contemporary business. According to some estimations (Batt et al, 2005, Brown et al, 2005, Gans et al, 2003) 70% of all business interactions are handled in contact centers. In the U.S., the number of all contact center workers is about 4 million or 2.5-3% of the U.S. workforce.

Creating business applications in contemporary Contact Centers is a very complex task. Indeed, typical CC applications (Anisimov et al, 1999) comprise Interactive Voice Response (IVR) scripts, routing strategies, call control, agent scripting, reporting, etc. Each of these functions has their dedicated tools and scripting languages and a CC application designer is required to be proficient in all of them.   More unfortunately, in most companies there are pools of experts in each of several quite distinct disciplines, making it extremely difficult to design end-to-end interaction management applications.

The heterogeneous structure of CC applications is a challenge also because many of the applications, such as routing strategies, are also strongly platform dependent. Since most of the leading contact center applications remain proprietary, it is quite common that applications developed for a specific contact center product cannot be easily transferred to another one.

A proven way of achieving application uniformity, platform independence, and simplification of the task of creating business applications is to employ XML-based standards and related technologies. XML is increasingly used as a basis for building applications in different vertical businesses. Good examples of XML-based standards for voice processing are the VoiceXML (VoiceXML, 2004) and Call Control XML (CCXML, 2005) protocols developed within W3C. They enable representation of any voice application as an XML document, and using VoiceXML and CCXML it is already possible to build simple CC applications involving only IVR (including automatic speech recognition capabilities) processing and simple call control and to represent them as a single XML document. The main advantages are obvious: uniformity, platform independence, and leveraging web technologies.

However, VoiceXML and CCXML do not address other important aspects of CC applications such as interaction workflow/service chain management (the process management task specialized on customer interaction management), interaction routing, scripting agent activities, reporting on agent performance and traffic management, using customer profiles, conducting
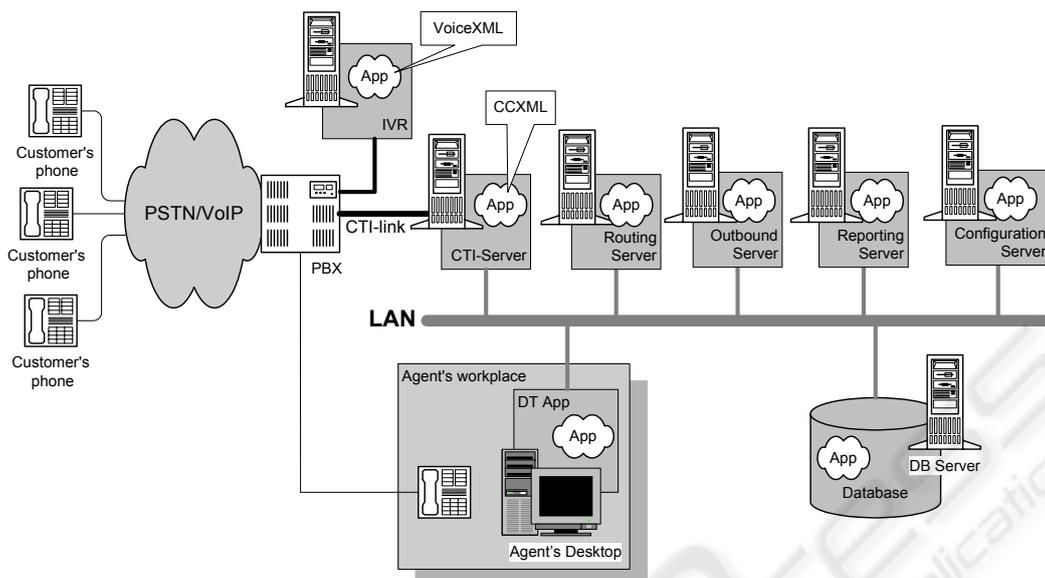
Figure 1: Contact Center environment.

outbound campaigns, and interactions that are VoiceXML and CCXML are, in fact, good examples of the strength and the limitations of the common approach to standards. This approach emphasizes bottom-up standardization, with each standard addressing a limited problem space; one typically solves the problem at hand and closely related problems when developing a new standard. The alternative top-down approach would be to design a purpose-built standard that could accommodate all of the elements needed for the

entire larger problem domain (in this case, end-to-end interaction management in a media-independent way). Such top-down standardization seems attractive but is probably unwieldy, and any failure to anticipate every possible contingency leads inevitably to serious flaws that make such standards likely to fail (in the sense that they do not achieve widespread acceptance and multiple implementations).

We introduce some ways of extending the VoiceXML and CCXML approach in order to provide coverage for additional important contact center functionality. We propose a methodology that is open to incremental extensions and that presents basic interaction management concepts such as platform and application, multi-script and multi-browsing, and interaction data processing without attempting a comprehensive top-down standard. The proposed methodology consists of a general XML-based interaction scripting framework called XContact, as well as a protocol for expressing the local interaction platform specifics (configuration,

conducted in media other than voice. rules/state machine and current state) called XPlatform.

The focusc is on main concepts and principles rather specific XML languages. The XML notation used in examples is self explanatory and serves for illustrative purposes only.

2 CONTACT CENTER ENVIRONMENT

Fig. 1 shows a typical structure of a contact center. The computing and telephony domains are connected through Computer Telephony Integration (CTI) technology (Chow et al, 2000) via CTI-Link and CTI-Server. The computing domain usually comprises several application services, each being responsible for particular functions of contact center operation, e.g. a routing server to find the most appropriate resource for inbound calls or an outbound server for outbound notification and generation of outbound calls to customers. The contact center database captures all information related to customers and customer interactions. Typical call processing is sometimes fully automated (self service), but often also requires providing assistance to the customer by Customer Service Representatives (CSR). Each CSR is represented in the contact center environment with a desktop and a phone connected to the switch.

Here is a typical scenario of inbound call processing: The inbound call is transferred to an IVR for collecting initial information. Sometimes the customer is fully served from within the application provided by the IVR; in other cases the IVR assists in "call steering" by asking the customer to specify what their service needs are. Then the call is transferred to a Router whose job is to select the most appropriate available CSR. The call is transferred to this CSR's phone. At the same time all collected information – potentially together with customer's historical information - is submitted to the CSR's desktop, and the CSR starts conversation, often guided during the conversation by prompts (displayed on the CSR's computer screen; this is called *agent scripting*).

As discussed above, currently only two components of the contact center application can be expressed in terms of standardized XML, i.e. the other functions require proprietary script languages and tools.

3 XML-BASED CONTACT CENTER FRAMEWORK

3.1 XContact and XPlatform

VoiceXML and CCXML were introduced to cover IVR and telephony call control functionality. Additional XML-based protocols are desirable to cover call routing, generation of outbound calls, agent scripting, multimedia interaction control, etc. We will denote the set of all of these XML languages (old and new) as XContact. Applications written in these languages will be called as XML scripts.

XContact contains also orchestration capabilities for building and executing complex applications composed of individual XML scripts. These orchestration facilities should include mechanisms for invocation of XML scripts, exchanging data between scripts, and other synchronization mechanisms, see Fig. 2. The XContact architecture assumes that platform-dependent components are clearly separated from XContact and its languages.

The term XPlatform is used to denote all platform-dependent components and data indicating that it shall also be extendable and expressed in XML-based notation. The actual XPlatform specification will usually vary across different contact center instances, while XContact and comprising languages do not depend on the actual platform. However, during execution an application

written in XContact has to be aware of the underlining platform and its specifics.

In summary XContact can be seen as a set of concepts, models and XML-based languages for the creation of complete contact center applications.
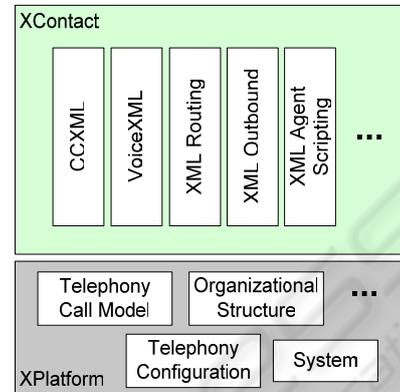

Figure 2: XContact and XPlatform.

3.2 Concept of Platform in VoiceXML/CCXML

One of the main advantages of VoiceXML and CCXML is application platform-independence. Voice applications written in VoiceXML/CCXML could run on any VoiceXML/CCXML platform because VoiceXML has been created in a similar way as HTML, and neither the VoiceXML language itself nor VoiceXML applications have any assumption about the platform structure.

However, with CCXML the situation is more complex. CCXML was designed to provide telephony call control as a complement to VoiceXML. CCXML uses a relatively simple call model that represents an abstraction of call models such as CSTA (CSTA, 2002), ECTF C.001 (ECTF, 1997), and JAIN (JAIN, 2001). It is assumed that CCXML applications could run on platforms with different call models. However, we cannot say that CCXML is fully platform-independent. Indeed, the CCXML call model specifies telephony events an application can receive from the platform. Some parameters of the events are platform dependent. For example, parameters such as ANI (Automatic Number Identification) and DNIS (Dialed Number Information Service), that are very important for contact centers applications, depend essentially on the underlining protocol.

One way to avoid such platform dependencies in CCXML is to explicitly specify the used call model as part of the platform. This specification may be

XML-based and contain the specification of at least all events with parameters. It also may contain a specification of requests and call model behavior in form of a state machine. The designer of a CCXML application shall be aware of the given call model and shall design the application based on its specification. Such approach allows one to not be limited by a specific call model but to use the most convenient and complete one that is appropriate for the particular platform elements. For instance, one can use the Genesys call model for voice applications intended to be executed on the Genesys platform - of course the specification of the Genesys call model has to be available to the application developer.

For end-to-end contact center applications the platform relationship is even more complex. Beside the telephony call model a contact center usually contains many other components that can hardly be unified. Examples are the organizational structure of CC personnel taken into account for finding the most appropriate CSR, interaction models for other media like e-mail and chat, etc.

3.3 Platform Structure

A contact center Platform may contain different parts responsible for operation of the contact center. Let us consider some important parts of the platform.

3.3.1 Interaction Platforms

The important parts of an "interaction platform" are components that are responsible for the management of "physical" interactions. There may be different platforms that are related to different media and different underlining network protocols. The interaction platform description typically includes the following parts:

- The interaction platform's configuration including description of configured objects and resources like available communication ports, directory numbers, devices, etc.
- The current state of the interaction platform including all active interactions existing within the platform. We will represent a collection of active interactions as an XML document containing a list of interaction elements with parameters like call legs or parties, their states, attached data (data that is tightly coupled to the particular interaction, such as customer account identification), and so forth (see Example below).

- A description of the interaction model that may include the set of events that may be issued by the interaction platform, the set of requests and associated parameters that can be used to control interaction processing, and the state machine that defines the behavior of the interaction platform. A designer of an XContact application should be aware of this model and his application must be compliant to the model.

3.3.2 Organizational Structure

The most important and expensive part of every real-world contact center is its workforce that comprises CSRs, managers, administrators, etc. The organization of a workforce in workflow systems is usually called organizational structure (CSTA, 2002). Each contact center has its own organizational structure that may be formed of branches, departments, groups, managers and CSRs. The designer of a CC application must be aware of the organizational structure in order to organize resource management in his application in an appropriate way.

3.3.3 System Data

The CC platform may maintain and expose to its applications a wide range of system data. An example is a time service providing applications with information about current time and day. Another example of system data is current values of service objectives of applications like average waiting time, abandonment rate, CSR occupancy, etc. This information can be used, for instance, for making routing decisions.

4 EXAMPLE: TOY CONTACT CENTER PLATFORM

For illustration purpose let's introduce an XML-based specification of a very simplified Toy Contact Center Platform (TCCP).

4.1 General Platform Structure

TCCP contains two main elements <voiceConfiguration> and <organizationalStructure> that will be described in more detail in the following sections.

```
<CCPlatform name="ToyContactCenter">
<voiceConfiguration name="">
. . . . . . . .
</voiceConfiguration>
<organizationalStructure>
. . . . . . . .
</organizationalStructure>
</CCPlatform>
```

4.2 Voice Configuration Platform

The TCCP voice configuration platform is described by the following XML document.

```
<voiceConfiguration name="ToyVoiceConfig">
  <switch name="SW1">
    <directoryNumbers>
      <dn num="1120" type="regular"/>
      <dn num="1121" type="regular"/>
      <dn num="1122" type="regular"/>
      <dn num="2239" type="routingPoint"/>
    </directoryNumbers >
    <calls>
      <call callID="0238AB82" type="voice">
        <leg type="external"/>
        <leg type="internal" dn="2239"/>
        <dnis value="8001234567"/>
      </call>
    </calls>
  </switch>
</voiceConfiguration>
```

The platform consists of two parts: configuration of directory number (DN) objects and description of all current interactions (calls). In the given example there are four configured DNs: 1120, 1121, 1122, and 2239. First three of them are regular DNs that represent CSRs' phones. The fourth DN is of type routing point that is associated with a routing strategy.

At the given time the platform contains only one active call specified by the element <call>. The call comprises two legs, one is associated with a customer and the other one with a routing point DN with a corresponding routing strategy. The call's DNIS number is contained in the <dnis> element.

4.3 Organizational Structure

TCCP has a very simple organizational structure consisting only of CSRs:

```
<organizationalStructure>
  <CSR Name="Mike First" status="Ready">
    <skill name="CustomerService" level="5"/>
    <dn num="1120"/>
    <occupancy value="0.96"/>
  </CSR>
  <CSR Name="John Second" status="Ready">
    <skill name="CatalogSale" level="5"/>
    <dn num="1121"/>
    <occupancy value="0.90"/>
  </CSR>
  <CSR Name="Petr Wise" status="Busy">
    <skill name=" CustomerService" level="2"/>
```

```
    <skill name="CatalogSale" level="5"/>
    <dn num="1122"/>
    <occupancy value="0.83"/>
  </CSR>
</organizationalStructure>
```

At the given time the TCCP has three CSRs. Each CSR has skills represented by the element <skill>. The first CSR, Mike First, has a skill related to customer service activity. The second CSR, John Second, has a skill related to catalogue sale activity. The third CSR, Petr Wise, is cross-trained and has both skills. Each CSR has a status representing his availability, and its value is given by the attribute "status". Note that the two first CSRs are in state "Ready" and can accept new calls. The third CSR is in state "Busy" and is already engaged in a call. Each CSR is associated with a directory number (element <dn>) representing his telephone. Moreover, each CSR has a statistic describing his occupancy, i.e. a ratio of busy time to overall work time within some time interval.

Note that attributes such as occupancy and status are highly dynamic and can be extracted from a separate server.

More generally, all data containing in XPlatform can be instantiated in different servers of the CC. This applies both to Organizational Structure and also to Voice Configuration, where e.g. <calls> are very dynamic.

5 MULTI-SCRIPT AND MULTI-BROWSER MODEL

Typical real world CC applications consist of a set of components, each component specifying the application logic that implements a particular aspect of CC application behavior. As a rule, these different components are represented in different ways, using different XML or proprietary protocols (for example, self service might be implemented using VoiceXML for voice and XMPP for IM self service). In this section we will consider how to organize multi-scripting in XContact.

5.1 Multi-scripting Example: VoiceXML/CCXML

Consider a VoiceXML/CCXML voice application that consists two types of components – CCXML component(s) describing call control logic and written in CCXML language and VoiceXML component(s) describing logic of voice dialog(s) and written in VoiceXML terms. We should stress that

both VoiceXML and CCXML have much in common. First, they are built on XML and have similar notation. Second, they have many common constructs like tags for working with variables, flow control mechanisms. However, this will not always be the case; in some cases, application elements implemented in very different ways still need to be orchestrated in order to deliver effective interaction management (for instance, routing logic using proprietary routing tools and establishment of a video session using SIP might need to be closely coupled).

We will refer to these application components written in different languages as scripts. Therefore a VoiceXML/CCXML application is built with the aid of two types of scripts – VoiceXML script(s) and CCXML script(s).

During run time different scripts are executed by different interpreters. For example, VoiceXML/CCXML applications are executed with the aid of two interpreters – VoiceXML browser and CCXML browser. The Fig. 3 illustrates this relation.
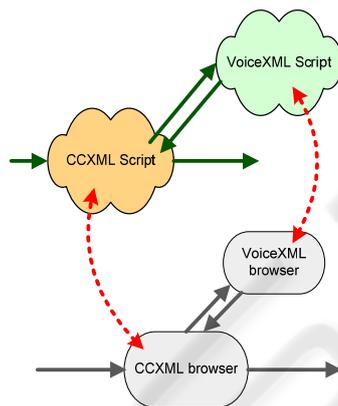

Figure 3: Scripts and browsers in VoiceXML/CCXML.

The invocation of VoiceXML script from CCXML one is organized by the element <dialogstart>. When the dialog is ended, the CCXML script receives a corresponding event "dialog.exit". Similarly, the CCXML script can invoke another CCXMl script using element <createccxml> whose completion is reported by an event "ccxml.exit".

5.2 Multi-scripting in XContact

Another example is a typical inbound voice application that includes additional components supporting CSR involvement like routing strategies and CSR applications.
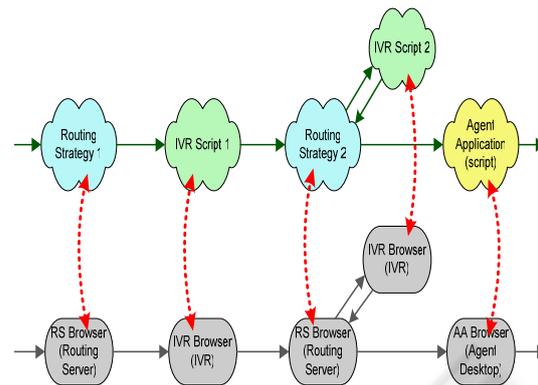

Figure 4: Scripts and browsers for real-world contact center.

This voice CC application comprises three types of scripts – routing strategy, IVR script and CSR desktop application. When an incoming call enters the CC, it is placed on a DN associated with Routing Strategy 1. Based on call attributes (e.g. DNIS) the call is distributed to an IVR that executes a corresponding IVR Script. The script controls a dialog with the customer collecting required information. Based on these collected data the call is transferred to Routing Strategy 2 that will find the most appropriate CSR. During execution the routing strategy may invoke another IVR script to collect additional customer information. And finally when the strategy determines an available CSR, the call is transferred to his/her telephone. This transfer is accompanied with initiation of agent's (CSR) scripting on his/her desktop assisting in conversation with the customer.

It is clear that when there are several script languages and scripts can invoke each other, we need a more general means for script invocation and termination. For such purposes we will use new elements like <startscript> indicating explicitly the type of script:

```
<startscript src="dialog.vxml"
        type="VXML" .../>
```

The completion of the script will be signaled by an event "script.exit" with returned parameters.

6 EXAMPLE OF APPLICATION: ROUTING STRATEGY

CC applications written in XContact languages will use data maintained in the platform. In this section we illustrate this using the example of a routing XML language. In Fig.3 the routing application is represented in a UML-like diagram.
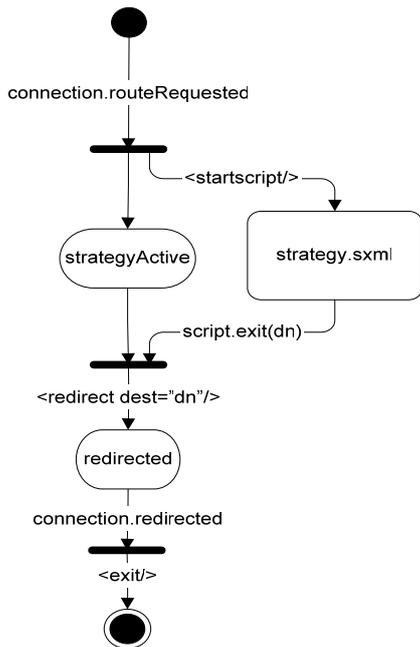
Figure 5: Application with routing.

The following strategy describes the routing of an interaction between two service types (activities): Customer Service and Catalogue Sale. It is assumed that these activities correspond to DNIS numbers 800-123-4567 and 800-765-4321 respectively.

```
<xstrategy name="ToyStrategy">
  <var name="dn"/>
 <!-- Rule for Customer Service activity -->
 <rule cond="//call[@dnis='8001234567']">
   <assign name="dn" value=
    "//CSR[/skill/@name='CustomerService']
        [@status='Ready']
        [min(@occupancy)]/@dn"/>
   <exit namelist="dn"/>
 </rule>
 <!-- Rule for Catalogue Sale activity -->
 <rule cond="//call[@dnis='8007654321']">
   <assign name="dn" value=
    "//CSR[/skill/@name='CatalogSale']
        [@status='Ready']
        [min(@occupancy)]/@dn"/>
   <exit namelist="dn"/>
 </rule>
</xstrategy>
```

The strategy consists of two rules, one corresponding to each activity. The attribute "cond" selects the appropriate call based on the value of the attribute "dnis" of calls. Note that all expressions in the strategy are represented in XPath notation (XPath, 1999). Applying the strategy to the call in the platform described above we realize that the call corresponds to the first rule. Next the strategy searches for the most appropriate CSR for the call.

The XPath condition for the variable "dn" returns a DN of the CSR who (i) has skill 'CustomerService", (ii) is in "ready" state, and (iii) has the minimum occupancy (compared to all available, appropriately skilled CSRs). Clearly, the third CSR will be selected. His DN "1122" is returned and the call is transferred to it using the call control request <redirect>.

7 RELATION TO W3C WORK

Recommendations VoiceXML and CCXML are being developed within the W3C voice browser working group. More general cases with different modalities are being developed within the W3C multimodal interaction group.

A VoiceXML environment is represented in Figure 6. The VoiceXML script is rendered by a VoiceXML browser controlling spoken dialog with a customer.
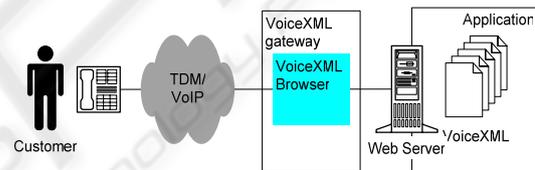

Figure 6: Scripts and browsers in VoiceXML environment.

Agent involvement in a contact with a customer can also be considered from web perspective, see Figure 7.
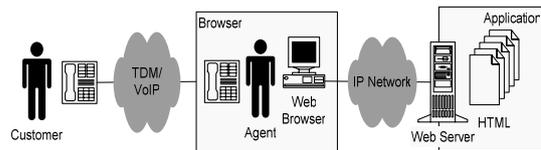

Figure 7: Agent as a voice browser.

One could think of it as the CSR playing the role of a browser "rendering" agent script dialog instructions written in HTML. Similar to VoiceXML an agent script specifies a dialog with a customer but in different terms. Moreover, CSRs usually use additional knowledge acquired during training process and sometime referred to as skills.

We can consider such environment as another modality or more strictly as another implementation of voice modality. The main difference here is that a CSR-browser should be found before starting the browsing session. Moreover, the CSR should have appropriate skills and be available (not busy). This

searching logic can be expressed in an XML-based form as routing strategy, see previous section.

The CSR environment can be considered as a special case of a W3C Multimodal architecture (Multimodal, 2006). In this architecture, VoiceXML and Agent scripts play the role of markup languages for modality components. CCXML and XML strategy are markup languages for controller and interaction management.

One can also use W3C State Chart XML (SCXML, 2006) as a good markup candidate for creation of a controller document that could organise combined use IVR and CSRs in one CC application.

8 CONCLUSION

Within this paper we introduced main concepts that we believe will be important for a comprehensive and consistent scripting of all contact center functions. In particular, the notion of a generalized XPlatform has been introduced representing the structure, capabilities and current state of the underlying real systems which are needed to handle interactions, and it is complemented by a standardized XContact specification that allows for end-to-end scripting of interaction management rules. A key element of the XContact approach is to allow usage of existing "bottom-up" protocols such as VoiceXML, CCXML, XMPP and many others within an orchestrated application framework that also provides missing elements such as routing strategy specification, outbound campaign rules specification, and so forth. Our future plans include the incorporation of applicable existing XML specifications and the development of XML languages for specific areas of contact centers that do not currently have coverage, as well as the more complete articulation of the XContact and XPlatform protocols.

REFERENCES

Anisimov N. *et al*. 1999. Formal Model, Language and Tool for Design Agent's Scenarios in Call Center Systems, *Proceedings of the 32nd IEEE Hawaii International Conference on System Sciences*, Hawaii, USA.

Batt R., Doellgast V., Kwon H. Service Management and Employment Systems In U.S and Indian Call Centres. Working paper 05-12. Industrial and Labor Relations School. Cornell University. July 2005.

Brown, L., Gans, N., Mandelbaum, A., Sakov, A., Zeltyn, S., Zhao, L. and Haipeng, S. Statistical Analysis of a Telephone Call Center: A Queueing-Science Perspective. (2005) Journal of the American Statistical Association, Vol 100, pp. 36-50.

CCXML. Voice Browser Call Control: Version 1.0. W3C Working Draft 29 June 2005. See http://www.w3.org/voice/

CSTA. Services for Computer Supported Telecommunications Applications (CSTA) Phase III. Standard ECMA-269. 5th Edition – December 2002.

ECTF. Call Control Model. Enterprise Computer Telephony Forum (ECTF) C.001. 1997.

Gans N., Koole G., Mandelbaum A. 2003. Telephone Call Centers: Tutorial, Review and Research Prospects, *Manufacturing and Service Operations Management,* vol.5, no.2, pp. 79–141.

JAIN Call Control, JSR 000021, 2001.

M. zur Muehlen. 2004. Organizational Management in Workflow Applications – Issues and Perspectives. *Information Technology and Management Journal.* Kluwer Academic Publishers, Vol. 5, No. 3, pp.271-291.

Multimodal Architecture and Interfaces. W3C Working Draft, April 14, 2006, http://www.w3.org/2002/mmi/

SCXML: State Chart XML: State Machine Notation for Control Abstraction, W3C Working Draft 24 January 2006. See http://www.w3.org/voice/

Sheng-Lin Chou, Yi-Bing Lin, 2000. Computer Telephony Integration and Its Applications, In *IEEE Communications Surveys & Tutorials*. vol. 3, no.1, pp.2-11.

VoiceXML. Voice Extensible Markup Language. Version 2.0. W3C Recommendation 16 March 2004. See http://www.w3.org/voice/

XPath. XML Path Language, Version 1.0, W3C Recommendation, 16 November 1999, See http://www.w3.org/TR/xpath