호 제어 마크업 해석기 개발 및 음성 대화 시스템과의 연동*

이경아(성신여대), 권지혜(성신여대),
김지영(성신여대), 홍기형(성신여대)

<차 례>

<Abstract>

Design and Implementation of a Call Control Markup Interpreter and Its Interaction with Voice Dialog Systems

Kyung A Lee, Ji-Hye Kwon, Ji-Young Kim, Ki-Hyung Hong

Call Control eXtensible Markup (CCXML) is a standard language that supports a call control of voice dialog systems such as VoiceXML based systems. CCXML allows developers to handle telephony calls in an easy way without deep knowledge about telephony networks and their switching systems.

We design and implement a call control markup interpreter. At the implementation, we use a Dialogic JCT-LS board, but, by designing a wrapping class for CTI (computer telephony board) features, the interpreter can easily adopt other CTI boards. We also design and implement event-based interaction scheme between the interpreter and voice dialog systems. For verifying the interaction scheme, we implement a simple voice dialog system.

* Keywords: CCXML, VoiceXML, Interactive voice dialog system.

* 이 논문은 산업자원부 지원으로 수행하는 21세기 프론티어 연구개발사업(인간기능 생활 지원 지능로봇 기술개발사업)의 일환으로 수행됨.

1. 서 론

호 제어 마크업 (CCXML, Call Control eXtensible Markup) [1]이란 VoiceXML 등과 같은 음성 대화 시스템의 가장 큰 응용분야인 전화망과의 연동을 위하여 전화 호 제어를 지원하도록 설계된 표준 문서이다. 호 제어 마크업은 XML문서로 개발자로 하여금 전화망과 전화 호에 대한 깊은 지식 없이도 손쉽게 전화 호를 제어하고 전화망 서비스와 음성서비스를 연동할 수 있도록 하는 기능을 제공한다. 이러한 특징은 전화망을 통한 음성서비스의 신규구축과 유지보수를 용이하게 한다.

기존의 전화망 서비스 시스템 구축을 살펴보면 전화 호를 제어하는 시나리오와 음성처리 및 관련 정보처리를 위한 알고리즘의 구현이 분리되지 않아 음성인터페이스와 전화망인터페이스 및 관련 정보처리를 모두 구현해야 하는 구조를 가지고 있다. 특히 구축이 완료된 후 시나리오 변동이 발생할 경우 개발자는 전체 프로그램 내용을 수정해야하는 추가적인 시간과 비용이 발생한다. 이에 반해 호 제어 마크업을 이용하면 전화망 서비스와 음성서비스 시나리오를 분리할 수 있어, 전화망 관련해서는 호 제어 마크업만을 사용하므로 서비스가 시작된 이후에도 간편하게 수정 및 유지보수가 가능하다. 이는 비용의 절감뿐 아니라, 효율적이고 동적인 전화 호 서비스를 제공할 수가 있다.

다른 특징으로, 호 제어 마크업은 VoiceXML과는 별개의 언어이다 [3]. 따라서 기존의 자동음성응답 (IVR)과 같은 음성 대화 시스템과도 통합될 수 있다. 이 또한 기존의 여러 음성 시스템과 연동하여 독립적으로 호를 제어할 수 있다는 점에서 큰 장점이다.

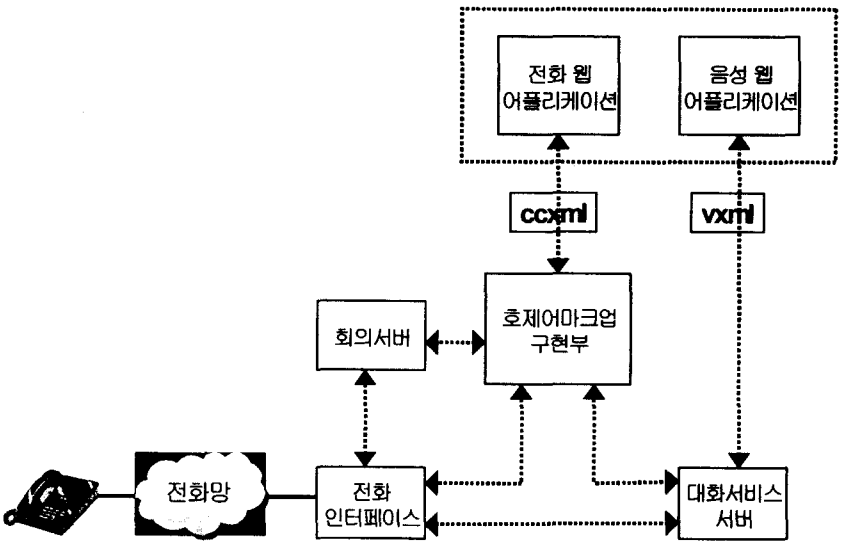
본 논문은 호 제어 마크업 문서를 해석하고 CTI (Computer Telephony Intergration) 기술이 적용된 DSP 보드를 이용하여 전화망을 직접 제어하는 해석기의 설계 및 구현에 대하여 기술하였다. 본 해석기는 Intel사의 CTI 전용 DSP보드인 Dialogic JCT-LS를 이용하여 구현하였으나 특정 CTI 보드에 종속적이지 않도록 시스템을 설계하였다. 특히, 본 해석기를 설계하고 구현함에 있어 기본적인 언어해석 이외에 가장 주안점을 둔 것은 호 제어 마크업 해석기 시스템과 다른 음성대화시스템과의 연동에 관한 것이다. 본 해석기는 두 시스템 간에 발생한 이벤트를 서로 주고받음으로써 호 제어 마크업의 해석기와 다른 음성 대화 시스템이 연계되어 동작할 수 있도록 구현 하였다. 이러한 기능을 구축하고 검증하기 위하여 별도의 간단한 음성 대화 시스템을 구현하여 검증을 마쳤다.

본 논문의 구성은 다음과 같다. 2장에서는 호 제어 마크업 시스템에 대해서 간단하게 소개를 하고, 3장에서는 해석기 설계에 관한 것으로 시스템 구성, 클래스 구성 및 이벤트 처리에 관하여 기술한다. 4장에서는 실제 구현된 시스템에 대해서 개발환경 및 구현된 예를 제시한다. 마지막으로 5장에서 결론 및 향후 계획을 기술한다.

2. 호 제어 마크업 시스템 소개

2.1 호 제어 마크업 시스템 개요 및 구조

서론에서 간략하게 소개한 바와 같이 호 제어 마크업은 전화 호를 제어하기 위한 XML문서이다. 호 제어 마크업 기반의 전화음성 정보시스템은 3가지의 주요한 구성요소를 가진다. 전화망으로부터의 전화 호, 음성대화시스템, 그리고 이 두 요소의 연결을 관리하는 호 제어 마크업 해석기 구현부이다. 전화망으로부터의 전화 호는 텔레포니 인터페이스를 이용하여 구현되며 특정 장치에 종속되지 않는다. 음성대화시스템 역시 VoiceXML 시스템뿐만 아니라 음성 미디어를 다루는 어떤 대화 시스템도 가능하다. 이 두 구성요소의 연결을 관리하는 호 제어 마크업 해석기 구현부는 문서를 해석하고 두 요소들과 연결하는 방법으로 이벤트를 이용한다. <그림 1>에서 호 제어 마크업 기반 전화음성 정보시스템의 시스템 구조를 보여준다 [1].



<그림 1> 호 제어 마크업 기반의 전화음성 정보시스템 구조도

2.2 호 제어 마크업 엘리먼트

호 제어 마크업 언어는 총 31개의 엘리먼트로 구성된다. 엘리먼트들은 그 성격에 따라 크게 다섯 가지로 분류될 수 있다. <표 1>은 호 제어 마크업 언어의 주요

엘리먼트들을 분류 별로 간략하게 소개하고 있다.

<표 1> 호 제어 마크업 엘리먼트

분류	엘 리 먼 트	내 용
문서의 흐름 제어와 실행	<ccxml>	CCXML 시작한다
	<if>, <elseif>, <else>	조건적 논리를 수행한다
	<fetch>	CCXML 파일을 미리 로드한다
	<goto>	실행을 새로운 주소로 이동한다
	<createccxml>	새로운 CCXML Session 생성한다
	<exit>	CCXML Session 실행의 종료한다
다이얼로그	<dialogstart>	다이얼로그 Session의 실행 시작한다
	<dialogprepare>	다이얼로그 실행을 위한 준비한다
	<dialogterminate>	다이얼로그 Session의 실행 종료한다
변수 및 구문	<assign>	변수에 값을 할당한다
	<var>	변수를 선언한다.
이벤트	<eventprocessor>	이벤트-처리 문의 영역
	<cancel>	CCXML event 타이머 취소
	<move>	다른 CCXML 세션으로 이벤트를 이동한다.
	<transition>	단일 이벤트처리 영역
	<send>	새로운 이벤트를 생성한다.
전화 기능 및 자원	<accept>	걸려오는 전화 수락한다.
	<createcall>	전화를 걸기 위한 새로운 호 형성한다.
	<createconference>	multi-party 오디오 conference 생성한다.
	<destroyconference>	multi-party 오디오 conference 종료한다.
	<disconnect>	전화 호 연결을 종료한다.
	<join>	두 음원을 연결한다.
	<redirect>	걸려오는 호를 새로운 호로 전달한다.
	<reject>	걸려오는 전화 호를 거절한다.
	<unjoin>	두 오디오 소스들 간의 연결을 종료한다.

2.3 음성 대화 시스템과의 관계

W3C Voice Browser Working Group은 호 제어 마크업이 VoiceXML과 같은 음성 대화 시스템의 호 제어부분을 보완 및 지원하기 위하여 제안된 언어임을 명시하고 있다 [1]. 호 제어 마크업은 전화 호를 제어하기 위한 언어이지만 호 제어 마크업만으로는 완전한 대화형 음성전화 서비스를 구현할 수 없다. <표 1>에 기술된 엘리먼트를 살펴보면 알 수 있듯이 대화형 서비스의 필수 요건인 전화 호 사용자와의 인터페이스에 관한 엘리먼트가 없기 때문이다. 완전한 대화형 전화서비스를 구축하기 위해서는 타 음성 서비스와의 연동이 반드시 이루어져야 한다. 이런 점에서 타 음성서비스와의 연동관계는 구현에 있어 매우 중요한 부분이다. 대표적인

음성 대화 시스템은 VoiceXML 기반 음성 시스템을 들 수 있다. W3C Voice Browser Work Group에서 제안하는 VoiceXML은 전화 호를 포함한 다양한 음성 인터페이스를 위한 표준을 제시하고 있다 [2][4][5]. 따라서 호 제어 마크업과 VoiceXML은 별개의 언어이지만 서로 보완관계이며 또한 독립적이다. VoiceXML로 구현되지 않은 음성 대화 시스템과도 같은 관계에 있다고 볼 수 있다.

호 제어 마크업 시스템과 음성대화 시스템과의 연동구현을 위하여 이벤트를 이용한다. 이는 독립적인 두 시스템 간에 발생한 여러 가지 사건과 정보를 비동기 이벤트를 발생시킴으로써 상대 시스템으로 전달한다. 이 이벤트 처리는 호 제어 마크업 해석기를 구현함에 있어 가장 중요한 부분 중 하나이다.

3. 호 제어 마크업 해석기 설계

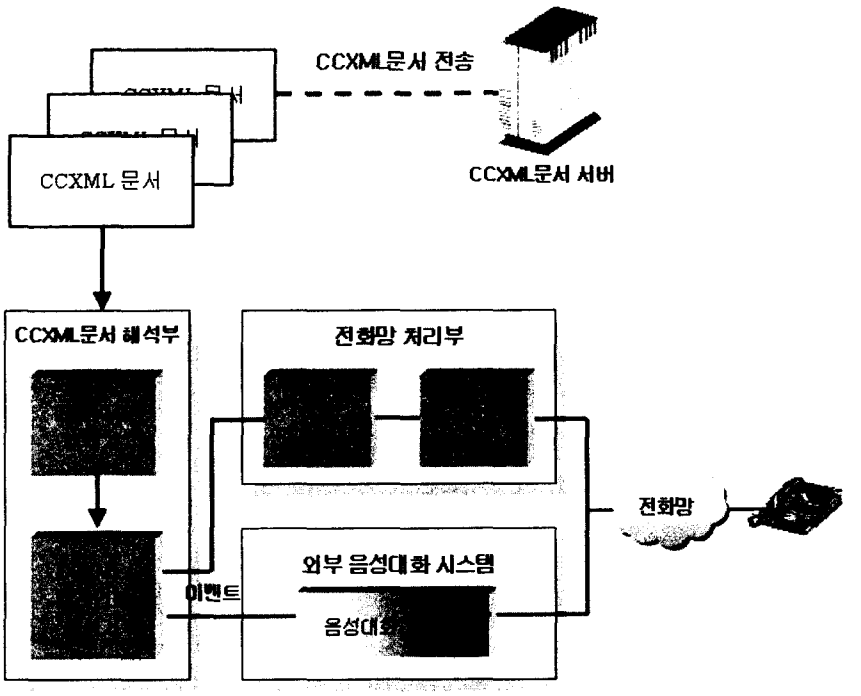
3.1 시스템 구성

<그림 1>에서 살펴본 바와 같이 호 제어 마크업 기반 전화음성 정보시스템은 전화망으로부터의 전화 호, 음성 대화 시스템, 그리고 이 두 요소의 연결을 관리하는 호 제어 마크업 해석기 구현부로 이루어진다.

첫 번째, 전화 호를 직접 제어하기 위하여 CTI용 DSP 보드가 필요하다. 그리고 이 하드웨어를 제어하기 위한 보드 API 구현부가 필요하다. 본 해석기는 Intel사의 Dialogic 보드를 사용하였으나 특정 하드웨어에 독립적인 구조를 갖도록 하기 위하여 그 상위계층으로 전화망 인터페이스를 따로 정의하였다. 따라서 다른 종류의 DSP보드를 사용하더라도 다른 부분의 수정 없이 보드 구현부만 변경하면 되도록 설계하였다.

두 번째, 외부 음성대화시스템은 테스트를 위하여 별도로 간단한 음성대화시스템을 구현하였다. 이 음성대화시스템은 지정된 채널에 사용자가 임의의 음성 명령을 내릴 수 있으며 종료시 호 마크업 해석기로 이벤트를 생성한다.

마지막으로, 호 제어 마크업 해석기는 CCXML 문서를 해석하기 위하여 일차적으로 XML 문서 해석기를 이용하여 호 제어 엘리먼트를 추출해 낸 후 엘리먼트에 따라 그 기능을 수행한다. 이 XML 문서 해석기는 IBM사에서 개발한 IBM XML4C를 이용하였다. 본 해석기에 관한 시스템 구성도는 <그림 2>와 같다.



<그림 2> 호 제어 마크업 해석기 시스템 구성도

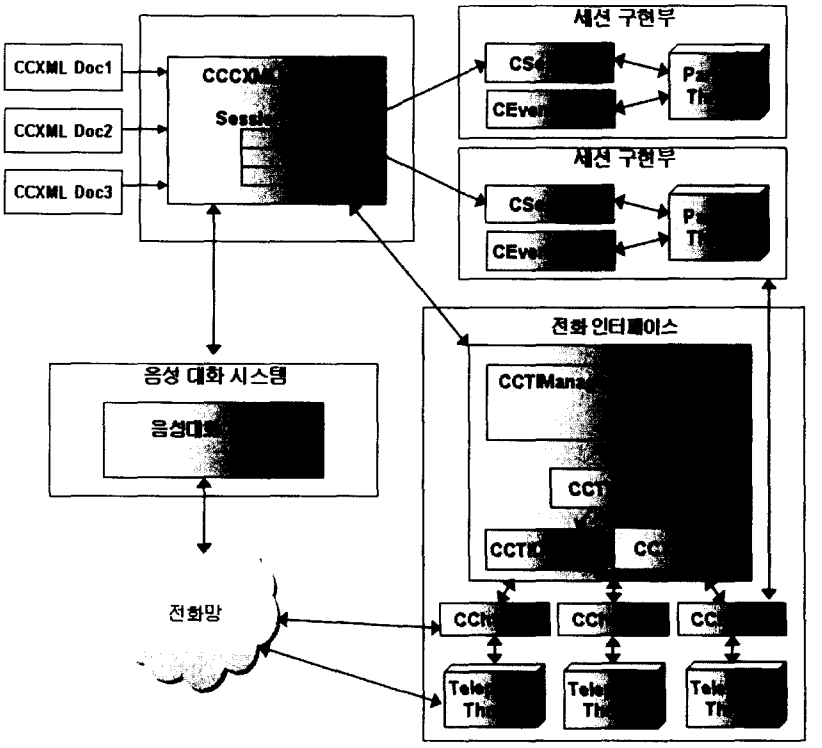
3.2 클래스 구성

호 제어 마크업 기반 전화음성 정보시스템은 세션 (session), 연결 (connection), 다자간 회의 (conference object), 그리고 음성 다이알로그 (voice dialog)의 4가지 객체로 구성된다 [1]. 세션은 하나의 CCXML 문서나 관련된 다수의 CCXML 문서로 구성되는 전화망 서비스이다. 연결은 실제 전화망에서 이루어지는 전화 호의 연결을 의미한다. 다자간 회의 객체는 다수의 미디어 스트림 (다수 화자의 음성 등)에 대한 믹싱을 담당한다. 마지막으로 음성 다이알로그는 호의 연결과 시스템, 또는 하나의 호와 다른 호의 연결을 기반으로 단방향 또는 양방향의 대화를 의미한다.

본 해석기는 객체지향기법으로 설계하여 각각의 기능을 가진 객체들을 클래스로 설계하였다. <그림 3>은 클래스 구성도이다.

우선 입력된 문서에 할당된 세션들을 관리하는 기능을 가진 CCCXMLManager가 전체 시스템을 관장한다. 시스템 내부는 크게 문서를 해석하는 세션구현부와 전화 호를 제어하는 전화인터페이스로 나뉜다. 첫 번째 세션 구현부에서는 세션을 CSession 클래스를 통해 구현하는데 세션별로 독립적인 처리를 해주기 위하여 각 세션별 파싱 쓰레드와 CEventQueue가 존재한다. 두 번째로 전화인터페이스는 CCTIManager가 전체 전화망관련 기능을 관장하는데 현재 연결된 채널들을 관리하

고 특정 CTI보드에 독립적인 인터페이스를 제공한다. 이를 위하여 Polymorphism을 이용하여 CTI보드에서 해 주어야 하는 일을 나타내는 클래스 CCTIBoard를 두었고 CCTIDialogic이나 CCTINMS 클래스는 CCTIBoard로부터 상속 받도록 구성하여 동적 바인딩이 쉽게 이루어질 수 있도록 하였다. 추후 CTI보드별 클래스가 추가되더라도 CCTIManager 클래스의 구현부를 수정하지 않아도 가능하게 된다. 다중 채널 사용을 위하여 각 채널을 CChannel클래스에 할당하였고 이 채널들 역시 독립적으로 이벤트를 처리할 수 있도록 전화처리 쓰레드를 두었다.



<그림 3> 클래스 구성도

3.3 이벤트 처리 방식

이벤트 처리는 호 제어 마크업의 가장 강력한 기능 중의 하나이다. 다양한 출처로부터 발생하는 호 제어 마크업 이벤트의 처리는 외부 음성 대화 시스템과의 연동에 있어 가장 핵심적인 부분이라 할 수 있다. 외부 음성 대화 시스템 또는 DSP 보드로부터 발생한 사건이나 정보는 비동기적 이벤트로 전달되는데, 이 이벤트를 처리함으로써 두 시스템간의 연동이 이루어지게 된다. 이벤트 처리를 위하여 <eventhandler> 엘리먼트를 사용하는데 이 엘리먼트만을 이용하여 이벤트에 접근할

수 있다. 각각의 이벤트에 대한 처리는 그 자식인 <transition> 엘리먼트를 사용하여 그 다음 기능을 기술하게 된다. <eventhandler>와 <transition> 엘리먼트를 구현하기 위하여 본 해석기에서는 각 세션별로 각각의 이벤트 큐와 파싱 쓰레드를 두었다. 이것은 세션별로 독립적인 이벤트 처리를 가능하게 해주고, 속도 향상 및 병목현상을 방지할 수 있다.

본 해석기는 W3C Voice Browser Working Group이 제시하는 Event Handler Interpretation Algorithm (EHIA)에 따라 구현하였다 [1]. EHIA는 다음과 같다. EHIA를 구현하기 위해서는 주 루프(loop)가 존재하는데, 먼저 이벤트 큐에 들어온 이벤트에 대하여 <eventhandler> 엘리먼트의 자식 노드인 <transition> 엘리먼트들 중에서 이벤트이름과 상태변수 값이 만족하는 <transition> 노드를 선택한다. 선택된 <transition> 노드가 있다면 그 하위 엘리먼트들을 순서에 따라 처리한다. 선택된 <transition> 이 없다면 그 이벤트는 제거한다. 이벤트 큐가 비어있다면 이벤트큐에 이벤트가 발생할때까지 무한정 기다리게 된다. 단 오류가 발생하거나 <goto> 또는 <exit> 엘리먼트를 만나게 되면 <eventhandler> 엘리먼트 처리를 종료하게 된다.

4. 해석기 구현

4.1 개발 환경

본 해석기의 호 제어 마크업은 CCXML Version 1.0(W3C Working Draft 11 January 2005) [1]을 기준으로 하였다. 멀티 쓰레드 기법을 이용하여 동시에 여러 문서를 해석하여 실행할 수 있도록 구현하였다. 본 해석기 구현은 4채널 보드를 이용하였으므로 4개의 각 채널에 사용자가 선택하여 문서를 할당할 수 있고 동시에 해석한 후 실행이 가능하다.

개발환경은 윈도우즈 환경을 기반으로 하였으며 CTI용 DSP보드는 Intel Dialogic D/41 JCT-LS(4 채널)을 이용하였다. XML해석기는 IBM사에서 제공하는 XML4C를 이용하였다[6]. 상세한 개발환경은 <표 2>에 명시되어 있다.

<표 2> 시스템 개발환경

분류	구성요소	세부내역	비고
하드웨어	CPU / RAM	Intel Pentium 5 / 512MB	
	CTI DSP 보드	Intel Dialogic D/41 JCT-LS	4채널
소프트웨어	XML 해석기	IBM XML4C	XML DOM Parser
	개발 언어	C++	
	Compiler	Visual C++ 6.0	
	운영체제	Windows XP Professional	

4.2 구현 예제

본 해석기의 시험은 CCXML Version 1.0에 제시된 제어 마크업 문서 예제를 참고하여 이루어졌으며, 그 중 한 가지 예제는 다음 <표 3>과 같다.

<표 3>실험에 사용된 호 제어 마크업 문서

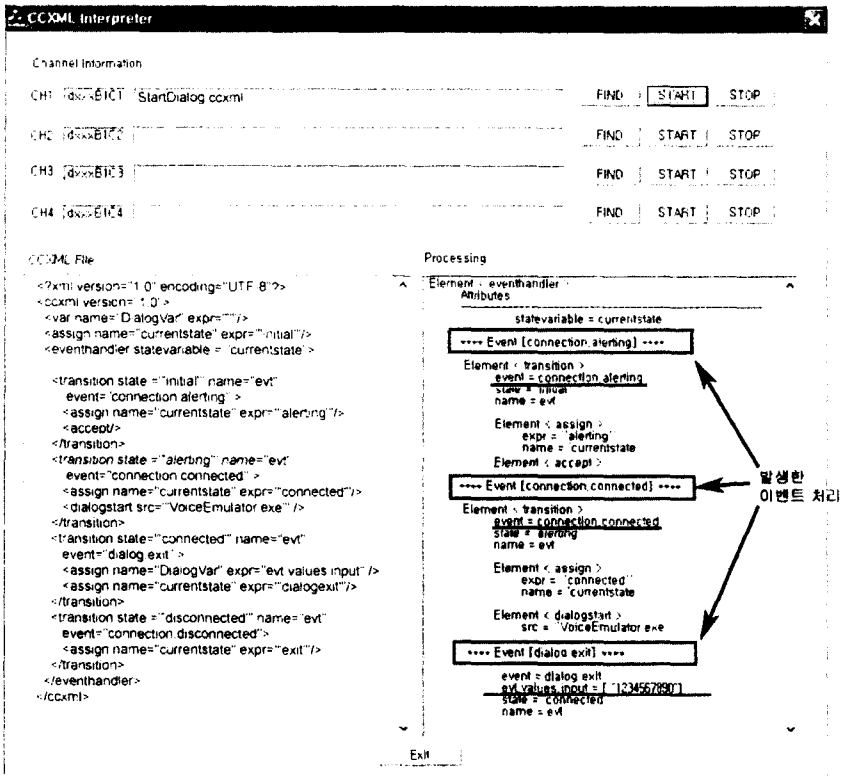
```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <ccxml version="1.0">
3.     <var name="DialogVar" expr="''"/>
4.     <assign name="currentstate" expr="'initial'"/>
5.     <eventhandler statevariable = "currentstate">
6.         <transition state ="'initial'" name="evt" event="connection.alerting" >
7.             <assign name="currentstate" expr="'alerting'"/>
8.             <accept/>
9.         </transition>
10.        <transition state ="'alerting'" name="evt" event="connection.connected" >
11.            <assign name="currentstate" expr="'connected'"/>
12.            <dialogstart src="'VoiceEmulator.exe'" />
13.        </transition>
14.        <transition state="'connected'" name="evt" event="dialog.exit" >
15.            <assign name="DialogVar" expr="evt.values.input" />
16.            <assign name="currentstate" expr="'dialogexit'"/>
17.        </transition>
18.        <transition state="'disconnected'" name="evt" event="connection.disconnected">
19.            <assign name="currentstate" expr="'exit'"/>
20.        </transition>
21.    </eventhandler>
22. </ccxml>
```

우선 문서해석을 시작하기에 앞서 CCCXMLManager의 인스턴스를 생성하고 모든 초기화를 작업을 한다. CCCXMLManager 클래스는 전체 시스템을 관장하는 클래스로 인스턴스는 한번만 생성된다. CCTIManager 인스턴스를 생성하고 받아들인 CTI보드정보에 따라 폴리모피즘을 이용하여, CCTIBoard 클래스를 특정보드에 해당되는 클래스 인스턴스로 생성하여 동적으로 할당한다.

위의 코드는 2번줄에서 <ccxml>엘리먼트로 시작하게 되는데 이때 CSession 클래스의 인스턴스를 생성하여 이 문서에 대한 실행이 끝날때까지 세션관리를 책임진다. 이때 사용자로부터 문서에 할당된 채널정보도 받게 되므로 CChannel클래스의 인스턴스를 생성한뒤 해당 채널을 할당한다. CChannel 클래스는 자신의 채널을 오픈하고 대기상태로 만든다. 5번 줄에서 <eventhander> 엘리먼트를 보면 속성중

상태변수로 currentstate가 지정되어 있다. 이것은 하위 엘리먼트인 <transition>이 각 이벤트를 처리할 때 현재상태의 값을 체크하기 위하여 사용된다. 위의 코드에서 <eventhander> 엘리먼트는 총 4개의 자식 노드를 가지는데 <transition> 노드당 하나의 이벤트를 처리하도록 되어있다.

6번 줄의 <transition>노드는 초기상태에서 connection.alerting 이벤트가 발생하면, 즉 전화 착신호가 발생하면, currentstate변수에 alerting값을 할당하고 전화 호의 수신을 허락한다. 최초 이벤트 발생은 CChannel에 연결된 쓰레드가 주기적으로 CTI 보드의 상태값을 체크함으로서 가능한다. 상태변화가 발생하면 CChannel 클래스는 자신를 관리하는 CSession에게 이벤트를 발생시켜 준다. CSession클래스의 이벤트 큐에 이벤트가 쌓이게 되면 Parsing Thread가 <transition> 엘리먼트중 일치하는 것을 찾아 실행을 하게된다. 10번 줄에 있는 <transition>노드는 connection.connected 이벤트를 받으면 "VoiceEmulator.exe" 이라는 음성대화 시스템을 구동한다. 음성 대화 시스템과의 연동에 관한 자세한 내용은 4.3절에서 자세히 언급할 것이다. 14번 줄의 <transition>은 음성 대화 시스템이 종료되었을 때, 그리고 18번 줄의 <transition>는 통화가 종료되었을 때 발생되는 이벤트를 처리한다. <그림 4>는 이 문서를 해석하고 이벤트를 처리하는 본 해석기의 실행 예이다.



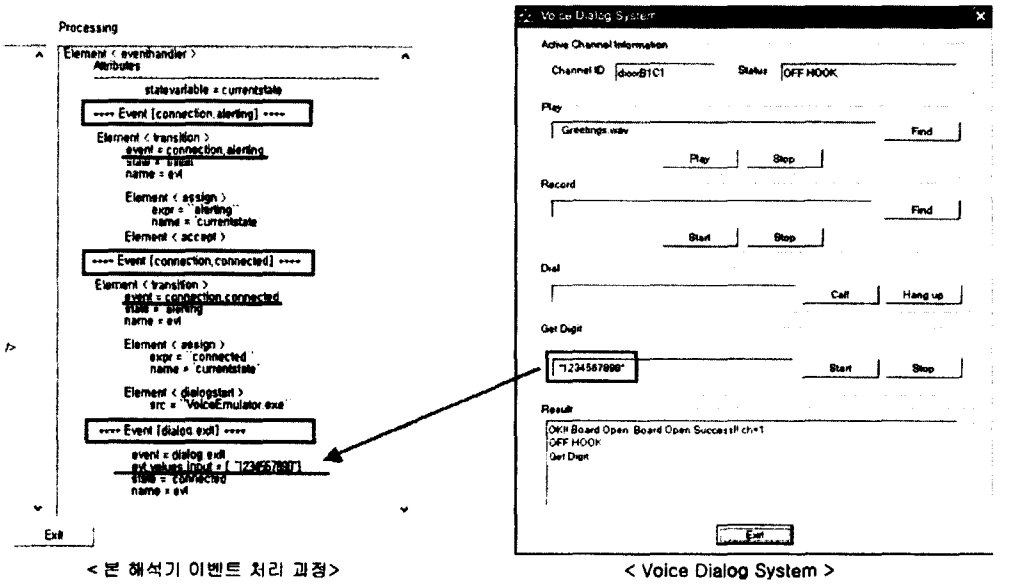
<그림 4> 호 제어 마크업 해석기 실행 예

보는 바와 같이 첫 번째 채널에 "StartDialog.ccxml"이라는 문서를 할당하면 프로그램 좌측하단에 문서 내용이 표시된다. 그리고 시작 버튼을 누르면 문서 해석이 시작되고 이벤트를 대기한다. 이 과정은 우측 하단에 표시되어 있다. 이벤트 connection.alerting가 발생하여 전화호수신을 허락하고 나머지 이벤트를 처리하였다.

4.3 음성 대화 시스템과의 연동 구현

앞서 언급한 바와 같이 호 제어 마크업 해석기 구현에 매우 중요한 부분을 차지하는 것이 외부 음성 대화 시스템과의 연동이다. 이 연동에 관한 구현을 시험하기 위하여 별도의 간단한 음성 대화 시스템(Voice Dialog System)을 구축하였다. 이 프로그램의 기능은 실행이 시작되면서 자신에게 할당되는 채널정보를 읽어서 전화망 채널을 열고 사용자에게 음성파일 출력, 녹음, 번호입력등 다양한 기능의 음성 대화 서비스를 제공하고 종료 시 본 해석기로 사용자가 입력한 정보를 이벤트와 함께 전송한다.

<표 4>의 13번 줄에서 <dialogstart src="'VoiceEmulator.exe'" /> 엘리먼트를 만나면 자신의 채널정보를 저장하고 Voice Dialog System 프로그램을 자동으로 실행시킨다. Voice Dialog System 프로그램이 실행되면서 채널을 열고 사용자로부터 음성 서비스 명령을 대기한다. 이 서비스 중 사용자가 숫자를 입력하고 Voice Dialog System을 종료하게 되면 본 해석기 시스템으로 dialog.exit 이벤트를 발생시키고 매개변수에 사용자가 입력한 값을 저장하여 함께 전송하게 된다.



<그림 5> 본 해석기와 음성대화시스템과 연동 실행 예

이 내용이 <그림 5>에 나타나 있는데 사용자가 전화기로 입력한 "1234567890"이라는 숫자는 Voice Dialog System이 종료 시에 발생하는 dialog.exit이벤트의 매개변수 중 evt.values.input에 저장되어 본 해석기로 전달되게 된다. 이런 방식을 이용하여 사용자와의 대화형 서비스가 가능하게 된다.

5. 결론 및 향후 과제

본 논문에서는 호 제어 마크업 해석기의 기본적인 구현뿐 아니라 호 제어 마크업 해석기 시스템과 음성 대화 시스템과의 연동에 초점을 맞추었다. 이를 위하여 별도의 간단한 음성 대화 시스템을 구축하였고 두 시스템간의 이벤트 발생 및 처리를 설계 구현하였다. VoiceXML 기반 시스템과의 연동은 현재 호 제어 마크업 시스템의 이벤트 처리를 완벽하게 지원하는 VoiceXML 해석기가 없어 이루어지지 못했으나, 본 구현과 시험은 향후 다양한 음성 대화 시스템과의 연동 구현에 있어 기초가 될 것이다. 향후 연구과제로는 다자간 통화관련 엘리먼트와 VoiceXML 기반 시스템과의 연동구현이 있으며, CCXML 문서의 작성을 도와주는 편집도구의 개발 등이다.

참 고 문 헌

[1] Voice Browser Call Control: CCXML Version 1.0
http://www.w3.org/TR/2005/WD-ccxml-20050111/W3C Working Draft 11 Jan. 2005.

[2] Voice Extensible Markup Language(Voice XML) 2.1
http://www.w3.org/TR/2004/WD-voicexml21-20040728/W3C Working Draft 28 July, 2004.

[3] 권하정, "대화형 음성서비스를 위한 호 제어 마크업 해석기의 설계 및 구현", 석사학위 성신여자대학교, 2003.

[4] 김정곤, "대량의 발신 호를 지원하는 음성 메시지 시스템", *말소리*, 제 49호, pp.77-94, 2004.

[5] 김학균, 김은향 외, "VoiceXML 기반 음성인식시스템을 이용한 서비스 개발", *말소리*, 제 43호, pp.113-125, 2002.

[6] IBM Corporation, "XML for C++ "
http://www-128.ibm.com/developerworks/xml/library/x-xercc/index.html13 Aug. 2003.

접수일자: 2005년 2월 10일
게재결정: 2005년 3월 15일

▶ 이경아(Kyung A Lee)
주소: 136-742 서울시 성북구 동선동 3가 249-1 성신여자대학교
소속: 성신여자대학교 미디어정보학부
전화: 02) 928-9997
E-mail: kaleehan@hotmail.com

▶ 권지혜(Ji-Hye Kwon)
주소: 136-742 서울시 성북구 동선동 3가 249-1 성신여자대학교
소속: 성신여자대학교 미디어정보학부
전화: 02) 928-9997
E-mail: jkwon@sungshin.ac.kr

▶ 김지영(Ji-Young Kim)
주소: 136-742 서울시 성북구 동선동 3가 249-1 성신여자대학교
소속: 성신여자대학교 미디어정보학부
전화: 02) 928-9997
E-mail: kj2112love@sungshin.ac.kr

▶ 홍기형(Ki-Hyung Hong)
주소: 136-742 서울시 성북구 동선동 3가 249-1 성신여자대학교
소속: 성신여자대학교 미디어정보학부
전화: 02) 920-7525
E-mail: khhong@sungshin.ac.kr