KSIAM IT series Vol.10, No.1, 71-82, 2006

컴포넌트를 기반으로 한 SOAP 구조
(A SOAP Architecture based on Components)

이희권, 서희석, 김희완
(Hee-Kwon Lee, Hee-Suk Seo Hee-Wan Kim)

요 약

SOAP은 XML과 HTTP 통신을 기반으로 하여 네트워크상에 존재하는 각종 컴포넌트간의 호출을 효율적으로 실현하기 위한 방법을 제시하는 규약이다. 본 연구에서는 XML웹 서비스와 .net을 이용하여 구성된 컴포넌트의 사용을 통해서 SOAP 구조의 전형을 보여주고 있다. 지식관리 시스템(KMS)의 구축을 통하여 다양한 컴포넌트 사용법과 XML을 통하여 시스템의 구조적인 모습을 구현하였다. 또한 기존에 있는 컴포넌트의 재사용을 통해서 객체 생성의 전형을 보여준다.

Abstract

SOAP presents a protocol to realize efficient call of each components. It is based on XML and HTTP communications and existed on the network system. In this paper, An example of the system is given through XML web service and components which are consist of .net structure. We built a Knowledge Management System which shows various uses of components. A pattern is proposed to create objects though reuse of existing components.

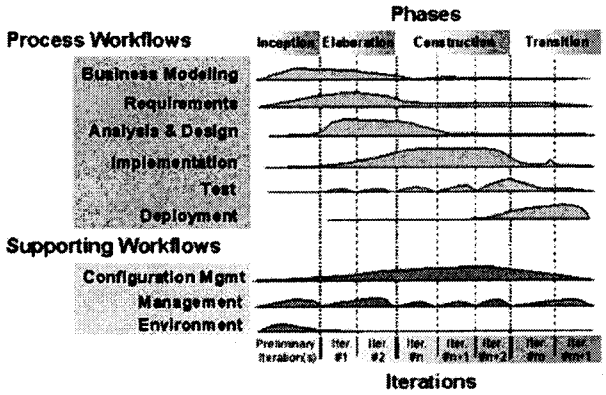
1. 서 론

SOAP(Simple Object Access Protocol)는 네트워크상에서 Client와 Service간에 메시지를 요청하고 이에 응답해주는 방법을 제공하는 것이다. 이러한 방식들은 기존의 RPC(Remote Procedure Call)라 묶여서 불려오던 것들이다. 따라서 SOAP은 RPC의 한 가지 방법이라 할 수가 있다. SOAP은 여러 Application Layer Protocol 들 중에 HTTP를 사용함으로써 여러 시스템간의 통신과 통합을 위한 좀더 단순하면서도 가벼운 메카니즘을 제공한다. 이외 HTTP를 사용하게 된 중요한 이유는 바로 방화벽에 제한을 받지

Key words : SOAP, XML, .net, reuse, components

않는 범용성 때문이라 말할 수 있다.

소프트웨어 개발 기반의 컴포넌트(Component-based software development)는 최근의 소프트웨어 개발의 모형이다[1]. CBSD는 소프트웨어 재사용과 소프트웨어 컴포넌트 기술의 연구로부터 시작했다(예제 [5] 과 [6]). 현재까지, 그것은 컴포넌트 모델, 컴포넌트 해설문, 영리목적에서 벗어난 컴포넌트, 분포된 오브젝트, 소프트웨어 구조, 소프트웨어 프레임워크, 그리고 디자인 패턴과 같은 소프트웨어 개발의 모형을 둘러싸는 문헌을 보고하는 집중적인 조사였다. 최근 몇 년간 몇몇 선도적인 조직과 전문가들에 의해서 컴포넌트를 기반으로 한 주목할 만한 소프트웨어 개발 접근법들이 소개되었다. 이들은 각기 소프트웨어 컴포넌트를 위해 고유한 개념과 이것을 다루기 위한 프로세스, 그리고 방법론을 지원하기 위한 도구를 함께 제공해 주고 있다. 이들 방법론은 전혀 새로운 접근법이라기보다는 공통적으로 기존의 방법론과 성공적인 경험을 토대로 진화하고 발전되어 만들어졌지만, 컴포넌트를 바라보는 각자의 시각과 이것을 통해 얻고자 하는 것에 있어서는 저마다 차이를 보이고 있다. 따라서, CBD 방법론을 도입하려는 사용자는 각각의 방법론이 갖는 특성을 세심하게 살펴봄으로써, 자신의 조직 문화나 프로젝트 성격에 가장 적합한 방법론을 선택할 필요가 있다.



[그림 1] RUP의 주기

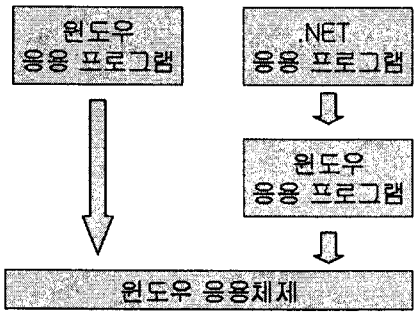
RUP(Rational Unified Process)는 소프트웨어 개발의 전체 생명주기를 지원하는 프로세스 프레임웍이다. 즉, 바로 적용하여 사용할 수 있는 방법론이라기보다는 다양한 유형의 소프트웨어 시스템, 도메인, 그리고 조직을 위한 프로세스의 프레임웍을 제공해준다. 따라서, 소프트웨어 개발 조직의 특정 요구사항에 맞게끔 프로세스를 조정할 수 있으며, 이것은 해당 프로젝트를 위한 Development Case에 문서화된다. RUP는 Objectory, OMT, Booch, 그리고 Rational의 접근법 등, 90년대 초반에 등장하였던 다양한 객체지향 방법론들을 통합하여 발전해왔다. 본 논문에서는 위에서 언급된 것처럼 컴퍼넌트의 사용과 그것을 재사용을 할 수 있는 .NET플랫폼을 이용하여 구현을 하였다. 본 논문에서는 시스템의 구현을 설명하고, 시스템을 이용하는 방법과 분석을 하였다.

2. SOAP 구현을 위한 구성 요소

2.1. .NET 플랫폼

윈도우 운영체제는 지난 수년 동안 발전과 지화를 거듭해 왔다. MS_DOS라는 16비트텍스트 모드 운영체제 위에서 실행되던 윈도우 3.1 시대부터 윈도우 XP 시대에 이르기까지 엄청난 기능의 향상이 있었고, 이에 따라 프로그래밍 환경도 변해왔다[3]. 그러나, 이러한 지속적인 발전은 두 가지 문제를 야기 시켰다. 첫째, 너무 기능이 많고 복잡해서 프로그래머가 이를 따라가기도 힘들어 졌다는 것이다. 둘째로, 급변하는 시대의 흐름을 따라가며 발전해오다 보니 프로그래밍 모델에 일관성도 떨어지게 되었다. 이러한 문제점에 대한 해결책으로 마이크로소프트가 발표한 것이 .NET플랫폼이다.

플랫폼이란 운영체제처럼 응용 프로그램에 인프라를 제공하는 시스템을 말한다. 하지만, .NET플랫폼은 독립된 운영체제가 아니고, [그림 2] 과 같이 윈도우 운영체제 위에 올라가는 런타임 환경이다. .NET 플랫폼을 이용하기 위해서는 운영체제에 .NET플랫폼을 따로 설치해 주어야 하는데, Visual Studio.NET을 설치하면 같이 설치되므로 우리는 .NET플랫폼을 따로 설치하지 않아도 된다. .NET 플랫폼은 GUI, 파일 입출력, 스레드 지원, 데이터베이스 접근, 보안, XML 지원, 웹서비스 등 폭 넓은 인프라를 응용 프로그램에게 제공한다.



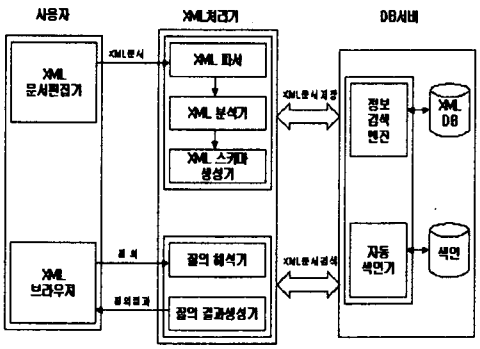
[그림2] .NET 플랫폼

2.2. XML

인터넷을 통한 거대 네트워크의 연결은 개인, 기업 내/기업 간, 나아가서 모든 것들을 인터넷이라는 수단으로 하나로 묶어 줄 수 있는 혁신적인 패러다임이다. 그러나, 아직 해결해야만 하는 문제점들이 존재하는데 이기종의 응용시스템의 통합, 서로 다른 포맷의 데이터 통합, 상황에 따른 웹페이지의 레이아웃 수정, 상호 연결 가능한 파일 등의 문제는 정보의 검색, 전자상거래, 데이터 처리와 공유 등의 부분에서 웹을 수용하고 활용하는데 문제점으로 대두 되었다. 이와 같은 문제들을 해결하기 위한 근본적인 대책으로 정보교환의 자동화에 문제의 초점이 맞추어 졌다[2]. 그리고 문제의 해결방안으로서 MetaData의 사용과 표준이 되는 Shared Context의 정립과 활용이 제시된다.

HTML은 디스플레이에 관해서는 다양한 편의를 제공하지만, 데이터 처리와 관련한 표

준 기반 방식은 제공하지 않다. 따라서 현재 HTML페이지와 마찬가지로 웹에서 구조적 데이터를 쉽게 이동시킬 수 있는 표준 표현(representation)을 바탕으로 하여, 데이터 표준인 XML은 새로운 분야에 두루 다양하게 사용되고 있다. XML은 마크업 언어가 아니라 마크업언어를 정의하기 위한 언어이다[2]. 즉 HTML과는 달리 Tag를 정의할 수 있고 데이터를 기술할 수 있다. 그래서 1998년에W3C에서는 인터넷에서 기존에 사용하던 HTML의 한계를 극복하고 SGML의 복잡함을 해결하는 방안으로써 XML을 발표하였다. XML은 웹상에서 구조화된 문서를 전송 가능하도록 설계된 표준화된 텍스트 형식의 마크업 언어로써 SGML의 Subset이며 SGML보다 훨씬 간결하고 인터넷에서 바로 사용가능한 문서를 표현하는 표준이다.

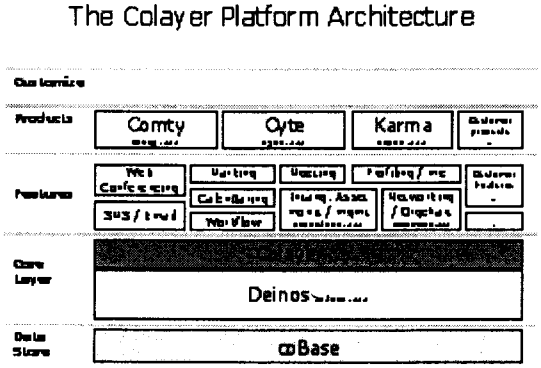


[그림 3] XML의 문서 처리 과정

3. 시스템의 구성

3.1. 시스템의 기본 구조

시스템의 기본 구조는 인터페이스에 기본을 둔다. Deinos의 중심부는 플랫폼의 다른 부분과 모듈로 통신을 한다. 하나의 규칙으로 모듈(co.dll)은 Deinos 인터페이스 위에서 통신을 하고, 상위의 모든 모듈은 Colayer 인터페이스 위에서 상호 작용을 한다. 이 시스템이 수행될 때는 co.dll을 반드시 레퍼런스하여 사용하여야 한다.



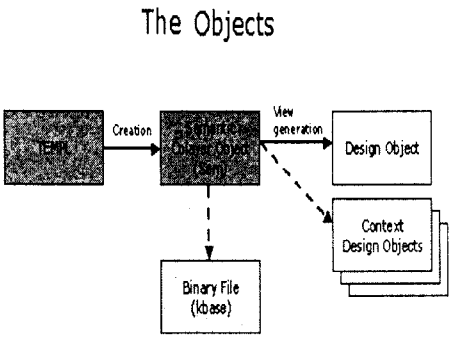
[그림 4] 시스템의 기본 구조

이 시스템의 모든 객체들은 XML 형태로 저장이 된다. 즉 모든 메시지나, 뉴스, 세션과 같은 이러한 형태를 가진다. 만약에 저장된 형태의 파일을 찾고 싶을 때에는, "coBase" 폴더 안에서 찾고자 하는 객체를 찾을 수 있다[4]. 이것의 환경에 대한 설정 개체는 단순한 XML 텍스트 파일에 저장이 된다. 이것은 개발 단계에서 선호되고 다루기 쉬운 단순한 형태이다. 만약에 보다 더 큰 형태의 데이터베이스가 요구가 될 경우에는 XML이 데이터의 저장하는 저장소로써 사용이 될 수 있다.

3.2. 웹 서버 구현

ATL은 웹 사이트 개발을 지원하는 서버로 스크립트와 C++를 섞어서 사용할 수 있기 때문에, 생산성과 성능이라는 것을 한꺼번에 표현할 수 있다[3].

ATL서버의 구조는 다음과 같다. 우선 ATL 서버는 인터넷 정보 서비스 IIS(Internet Information Server)와 함께 동작한다. 본 시스템의 경우, IIS5.0 Window 2000에서 제공되는 서버 서비스를 이용하였다. 서버의 구성은 |SAP| 확장 DLL, SRF(Server Response File) 및 태그 처리기 DLL로 구성되어 있다. IIS는 클라이언트로부터 들어온 요청을 ISAPI 확장 DLL로 넘겨준다. ISAPI확장 DLL은 동적 링크 라이브러리의 일종으로, 클라이언트가 보내온 HTTP 요청을 받아들이고 그에 대한 HTTP 응답을 내 주는데 필요한 기능을 수행한다. ATL 서버를 이용하면 ISAPI확장은 DLL이 자동으로 만들어 지므로, 이 부분은 우리가 크게 신경 쓰지 않아도 된다[3].



[그림 5] 객체의 조직

SRF 파일은 태그 처리기 DLL과 연동할 수 있는 특별한 태그를 가진 HTML 문서로, 출력 될 웹 페이지의 모양을 결정한다. 태그 처리기 DLL은 SRF 파일의 특정 태그를 처리하여 동적으로 출력될 웹 페이지를 만들어 낸다.

```
[ tag_name(name="Output") ]
HTTP_CODE OnOutput(void)
{
  int x= 5, y = 10;
  m_HttpRequest.Write(x);
  m_HttpRequest.Write(" +");
  m_HttpRequest.Write(y);
  m_HttpRequest.Write("     =
");
  m_HttpRequest.Write(x+y);

m_HttpRequest.Write("<br>");
  return HTTP_SUCCESS;
```

〔예제 1〕 예제 소스

위의 예제와 같이 Write 함수를 이용하여 출력하는 기능을 구현할 수 있다. HTTP 프로토콜은 Connectionless 프로토콜이다. 즉, 사용자가 웹 서버에 접속하고 있는 동안 서버와 클라이언트 사이에 연결이 지속되는 것이 아니라, 실제 데이터를 주고받는 동안만 연결이 되고 곧바로 연결이 끊어진다. 이러한 특성은 웹 서버로 하여금 많은 클라이언트를 수용할 수 있게 해 주었고, 웹이 크게 발전할 수 있는 원동력으로 작용했지만, 페이지 간의 연속적인 작업을 하기 위해서는 서버 측에서 세션 정보를 관리해 주어야 한다.

3.3. XML 형태의 구성

채널은 구독자에게 자동으로 푸시 될 수 있는 웹 페이지의 집합이다. CDF(Channel Definition Format)는 채널을 정의하기 위해서 마이크로소프트에 의해서 개발된 XML 어플리케이션이다.

채널은 웹 사이트의 중요한 정보가 변경된 사실을 사용자들에게 자동적으로 공지할 수 있다. 이 방법은 웹 캐스팅(webcasting)이나 푸시(push)라고 한다. 현재 인터넷 익스플로러는 CDF를 구현하는 유일한 주요 브라우저이다. 파일은 웹 사이트의 HTML 문서로부터 분리는 되어 있지만, 링크는 되어 있는 XML 문서이다. CDF 문서는 독자와 사이트의 컨텐츠 간의 연결을 위한 파라미터를 정의한다. 데이터는 푸시를 통해서 전달이 될 수 있다. 이것은 직접 사이트를 돌아다니면서 능동적으로 정보를 찾는 사람, 보통의 브라우징과 같은 웹 브라우저에 그들의 페이지를 로드하도록 선택하든지 업데이트 정보를 업데이트 하도록 할 수도 있다〔7〕.

직접 XML 어플리케이션을 개발할 때 프로젝트에 표현하고자 하는 데이터의 흐름이나 형태를 정돈할 필요가 있다. 이 과정은 엘레먼트 목록 작성하고, 기본 엘레멘트를 구분한 후에 엘레먼트의 관계를 구분하는 단계로 되어 있다. 본 논문에서는 웹 페이지에 나타내주는 것을 design폴더에 저장을 하였고, 그것의 데이터를 표현하는 것을 temple 폴더에 저장하여 표현을 하였다. 웹 페이지에 나타나는 객체를 표현하기 위해서 〔예제 2〕와 같이 표현을 하였다. 이 예제는 todo 객체의 실제적인 사용을 예제로 들어서 나타낸 것이다. 디

자인의 시작은 아래와 같이 〈DESIGN〉 태그로 시작을 하여서 〈/DESIGN〉 태그로 종료를 한다. 그리고 각각의 디자인에 들어가는 HTML코드는 〈HTML〉〈![CDATA[로 정의를 한 다음]]〉〈/HTML〉로 HTML 코딩을 마무리 짓는다. 이 태그 안에서 웹 페이지에 나타나는 HTML 부분을 표현할 수 있다. 여기에서 〈CO〉 태그를 볼 수 있는데 이것은 .Net 플랫폼에 있는 컴포넌트를 호출 하는 것이다. 서버 사이드에서 활동을 한다. 모두 실행 자체는 서버에서 되고 그 결과만 클라이언트(웹브라우저: MS 익스플로러, 모질라 등)에게 전송된다.

```
<DESIGN>
        <ID>todo</ID>

        <INHERIT>
                <HTML><![CDATA[

<table width=100% border=0 cellpadding=1 cellspacing=2 shuttle=Done>
<tr>
 <td rowspan=2 width=]]><CO>sem_indent</CO><![CDATA[% ]]><CO>sem_access</CO><![CDATA[></td>
 <td valign=top>
  <a href=http://colayer.com/help  target = _blank>
  <table width=24 height=24 border=0 cellpadding=0 cellspacing=0 bgcolor=#ffffff title=Open Todo>
        <tr>
          <td bgcolor=]]><CO>comty.todo_bleach</CO><![CDATA[ width=12 align=center><font style="font-weight:
          bold; font-size:9px; color: white"; font-family: Verdana,Arial;>!</font></td>
          <td width=12></td>
        </tr>
        <tr>
          <td colspan=2><font style="font-size: 9px; font-family: Arial"; color=999999>ToDo</font></td>
        </tr>
   </table>
   </a>
 </td>
 <td width=100%><b>]]><CO>sem_title</CO><![CDATA[</b><br>
        ]]><CO>sem_info</CO><![CDATA[
 </td>
</tr>
<tr>
 <td colspan=2>
        ]]><CO>pos_noadd<p>box</p></CO><![CDATA[
        ]]><CO>sem_txt</CO><![CDATA[
 </td>
</tr>
</table>
]]></HTML>
        </INHERIT><PARENT><LIB><IDREF>design</IDREF></LIB></PARENT>
</DESIGN>
```

〔예제 2〕 Design의 예

이것을 호출하는 레퍼런스를 만들 때에는, Co.cpp화일에 다음과 같은 정의를 필요로 한다.

CoEngine::add(객체이름,(void*)객체경로);
Co.h 헤더 파일에다 네임 스페이스를 만들고 객체의 메소드를 정의한다.

```
namespace cockpit
{
    void 함수 이름(파라메터);
}
```

〔예제 3〕은 실질적인 데이터를 저장하는 XML의 경우 Templ 폴더에 design에서 생성된 값들을 그 형태에 맞게 값들을 XML 트리의 데이터 형태로 저장을 한다. Templ을 정의할 때는 〈TEMPL〉태그를 처음에 정의를 하고 마지막에〈/TEMPL〉 태그로 마무리 한다. 디폴트의 값은 하나의 텍스트 박스로 정의를 하였고, 텍스트 에어리어나 콤보박스, 체크박스의 경우에는 〈TYPE〉의 태그를 이용하여 필요한 형태를 불러서 사용한다. 〈DESCRIPTION〉은 각 섹션의 제목을 정의하는 태그이다.

```
<TEMPL>
  <HISTORY>
    <CREA>
      <DATE>2003-04-05T00:00:00</DATE>
    </CREA>
  </HISTORY>
  <UPDATETREE>
    <DATE>1970-01-01T00:00:00</DATE>
  </UPDATETREE>
  <INHERIT>
    <DESIGN>
      <DESIGN>
        <IDREF>todo</IDREF>
      </DESIGN>
    </DESIGN>
  </INHERIT>
  <ID>todo</ID>
  <TODO>
    <C>
      <NAME>
        <_DESCRIPTION>Title</_DESCRIPTION>
      </NAME>
      <TXT>
        <_DESCRIPTION>Description</_DESCRIPTION>
        <_TYPE>TEXTAREA</_TYPE>
      </TXT>
    </C>
  </TODO>
  <PARENT>
      <LIB>
        <IDREF>templ</IDREF>
      </LIB>
  </PARENT>
</TEMPL>
```

〔예제 3〕 Temple의 표현

3.4. 클라이언트 사이드 표현

HTML은 정적으로 단순히 원하는 정보를 텍스트로 보여주기 때문에 동적인 사이트를 보여주기 위해서 다양한 방식의 스크립트언어가 나왔다. 그 스크립트가 클라이언트(웹 브라우저)에서 해석되어 작동하느냐, 아니면 서버에서 해석되어 작동하느냐에 따라 클라이언트 사이드 스크립트와 서버 사이트 스크립트로 나뉜다.

클라이언트 사이드 스크립트에는 DHTML, XML, VRML, XHTML, 자바스크립트, VB스크립트, 애플릿 등이 있다. 애플릿은 클라이언트에서 처리해야 하는 다양한 로직을 구현하는 데 사용된다. 특히 브라우저 화면의 세심한 컨트롤을 위해서라면 이러한 기술은 필수적이다. 앞에서 다루었던 design에서 보여 진 것처럼 HTML의 코드가 XML에서 인식이 되는 것을 보았다. 그러나 클라이언트의 언어인 자바스크립트, VB스크립트 같은 것은 철저한 클라이언트 상에서 작동이 되는 것이기에 먼저 스크립트 언어를 웹 페이지에 인식시키기 위해서 분리된 스크립트 형식을 만들었다.

〈J〉

```
〈ID〉organizer〈/ID〉
〈NAME〉form script〈/NAME〉
〈PARENT〉
  〈JS〉〈IDREF〉lib〈/IDREF〉〈/JS〉
〈/PARENT〉
〈CODE〉〈![CDATA[
function arcchecker(date,gid)
{
        ,,,,,,
        ,,,,,,
}
]]〉〈/CODE〉〈/J〉
```

이와 같은 형식으로 〈J〉라는 태그가 이용되면 안에 스크립트 언어를 정의할 수 있고, ID를 주어서 ID를 사용하는 서버상의 컴퍼넌트 페이지에 스크립트 언어를 사용하는 것이다.

4. 실험 결과

〔예제 4〕는 컴포넌트를 이용하여 Organizer에서 사용되는 링크를 표현하는 객체의 소스 코드이다. 이곳에서 앞에서 수행 id 및 현재의 위치를 가지고 오는 것을 정의한 것의 표현 방법이다.

```
Dtask *task = new Dtask();
```

이것은 현재 위치에서의 내부의 특정한 객체를 생성하는 임무를 주는 것이다. 이것은 특정한 ID를 가진다. 〈CO〉태그에서 이루어 졌던 coFuntion 이 호출이 될 경우, 하나의 파라메터로 id를 주어야 한다. 그리고 DArg를 통해서 현재의 페이지에 접근을 할 수 있다. DArg의 객체를 통해서 페이지 주소에 나타나는 상대적인 위치인 id를 얻을 수 있다. 그리고 DSem 컴포넌트를 통해서 페이지 정보를 담고 있는 XML파일의 특정 노드에 접근을 하여 저장된 정보를 불러들일 수 있도록 만들어 진 것이다.

```
void printOrgLinks(coint tid)
        {
        DTask * t = new DTask(tid);
        DArg   * arg = new DArg(tid);
        DSem * orgsem = new DSem(arg->iid());
        DSN * DisplayStatusnode = orgsem->xml(tid)->node("DISPLAY/STATUS");
        cos * Dstatus = DisplayStatusnode->child()->name();
        cos * table = new cos("");
             try
                    {
            table->cat("</td></tr></table></td><td align=center valign=middle width=30% bgcolor=#ffffff>
                                <table cellpadding=4><tr><td id=Archlink>");
            if(Dstatus->cmpi("CURRENT")==0)
               table->cat(new cos("<font class=handlink onclick=organizer.expandlink(W""),
               new cos(orgsem->gid()),new cos("W",W""),
               new cos("Archive"),
               new cos("W");><b><u>View archived Content</u></b></font><font id=__"),
               new cos(orgsem->gid()),new cos("_Archive></Font></td></tr></table>"));
            else table->cat(new cos("<font class=handlink
               onclick=win.get(W"karma.organizer.ChangeDStatus&1+"),
               new cos(orgsem->gid()),
               new cos("+W");><b><u>View current Content</u></b></font></td></tr></table>"));
            table->cat(new cos("<table cellpadding=4><tr><td><font class=handlink
                onclick=win.organizer.expandlink(W""),new cos(orgsem->gid()),
                new cos("W",W""),new cos("EventW");><b><u> Create a new Event
                </u></b></font></td></tr></table><font id=__"),
                new cos(orgsem->gid()),new cos("_Event></font><table cellpadding=4>
                <tr><td><font class=handlink onclick=win.organizer.expandlink(W""));
            table->cat(new cos(orgsem->gid()),new cos("W",W""),new cos("Invite"));
            table->cat(new cos("W");><b><u> Send an Invitation </u></b></font></td></tr></table>
               <font id=__"),new cos(orgsem->gid()),new cos("_Invite></font><table cellpadding=4>
               <tr><td><font class=handlink onclick=win.get(W"karma.addressbook.getaddressbook&"),
               new cos(DUtil::gid(arg->iid())),new cos("W");><b>"),
               new cos("<u> Get Addressbook ></b></font><br/><br/></td></tr></table></td></tr></table>"));
            arg->design(tid,table);
                       return;
                    }
         _catch( Err::breakThread, "Error in organizer::printOrgLinks" );
             arg->design(tid,"");
        }
```

〔예제 4〕 Organizer 링크를 표현하는 코드

5. 결론 및 향후 과제

다른 구조 관리자는 시스템의 컴포넌트가 프로그램 된 근원적인 시스템으로부터 아주 별개의 환경을 제공한다. 예를 들어 NMS는 활동하는 데이터베이스를 구조 정보를 유지하기 위해 사용한다. 시스템의 자동화된 재구성은 데이터베이스를 위한 규칙에 의해 지배된다. 코닉(Conic)은 분리된 필수적인 구조 언어와 자동화된 재구조를 위한 오직 아주 제한된 지원을 제공한다. 그러나 이 프로젝트는 구조 관리 임무로의 정도성의 방법에서 기존의 객체 지향언어로써 연장한다. 객체 지향 프로그램에 사용되는 구조 기술은 이렇게 구조 관리에서 돕기 위해 가능하다. 방해물은 구조 시스템과 필수적인 프로그램 시스템 사이의 부적당함은 최소화 한다.

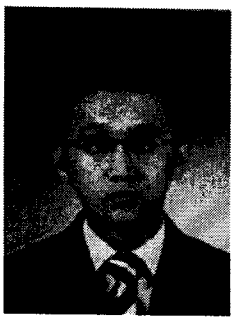
이것은 직접적으로 구조 관리 정보와 관리되는 객체들과 연계된다. 구조관리 활동의 분배는 이렇게 객체 그 자체의 분배의 자연스러운 결과이다. NMS에서 비슷한 분배의 단계

는 집중시키는 데이터베이스로부터 분배된 것으로 추가적인 머리위의 이동을 수반한다. MAD[8] 네트워크 관리로의 접근은 분배하는 관리 활동을 위한 기계 장치를 제공한다. 그러나 관리 활동의 분배가 기본 시스템의 기계장치 분배와 반복하지 않기 때문에 단순성이 부족하다.

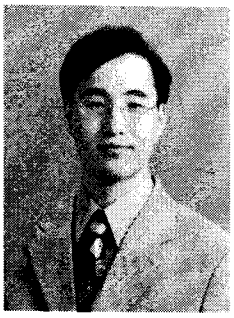
구현한 예제 데이터베이스는 XML를 사용하여 데이터의 형태를 만들었다. 그러나 SQL과 같이 Query가 존재하지 않는 XML을 데이터베이스로써 사용하기에는 한계를 가지고 있었다. XML의 경우 데이터를 검색하기 위해서 Loop 문을 이용하여서 비교 대입 후 데이터를 찾는다. 이것은 상당히 비효율적이므로 앞으로의 과제로 대 용량의 데이터가 필요한 부분에서는 전문적인 데이터베이스 시스템과의 연동을 통한 구현이 필요하다.

참 고 문 헌

1. M. Aoyama, Component-Based Software Engineering: Can it change the Way of software Development?
2. http://my.dreamwiz.com/aphise/, 안항준 XML.
3. Visual C++.NET programing Bible, 2nd Edition, 김용성
4. The way of Programming Colayer, Macus Hegi, 2003
5. H. Mili, F. Mili, and A. Mili Reusing Software Issues and Research Directions, IEEE Transaction on Software Engineering, vol.21, No.6, pp528-56, June, 1995.
6. F. Yang, H. Mei and K. Li. Software Reuse and Sofrware Component Technology, Acta Electronica Sinica, vol. 27, no. 2, pp. 68-75, Feb. 1999
7. XML BIBLE -Elliotte Rusty Harold
8. German Goldszmidtt, Yechaim Yemini, and Shaula Yemini. Network management by delegation - the MAD approach. Proceedings of the 1991 CAS Conference, pages 347-359, 1991



이 희 권(Hee-Kwon Lee)
· 2003. 3.~2004. 2 인도 Pune 대학교 I2IT과정 이수
· 2004. 2.~2004. 11 Colayer 연구소 연구원
· 2006. 2. 삼육대학교 컴퓨터과학과 졸업(이학사)
· 현재 시티은행 근무(인터넷뱅킹 시스템 개발 및 관리)
· 관심분야 : 웹서비스, XML, 웹데이터베이스



서 희 석(Hee-Suk Seo)
· 2000. 2. 성균관대학교 산업공학과 졸업 (공학사).
· 2002. 2. 성균관대학교 전기전자 및 컴퓨터공학부 졸업(공학석사).
· 2004. 3. 한국 정보 감리 평가원 (선임 연구원).
· 2005. 2. 성균관대학교 전기전자 및 컴퓨터공학부 졸업(공학박사).
· 2005. 3. ~현 한국기술교육대학교 인터넷미디어공학부 전임강사.
· 관심분야 : 네트워크 보안 시뮬레이션, 지능형 시스템, 분산 에이전트.

김 희 완(Hee-Wan Kim)
· 성균관대학교 전기전자 및 컴퓨터공학부(工學博士)
· 정보관리 기술사
· 삼육대학교 컴퓨터학부 부교수
· 관심분야 : 컴퓨터 및 네트워크 보안, XML, 분산 DB, 웹 데이터베이스