XML 기반 워크플로우 메니지먼트 시스템의 설계 및 구현

오명은[O] 윤석현[O] 한상용
중앙대학교 컴퓨터공학과
{jmania[O], lazecool[O], hansy }@archi.cse.cau.ac.kr

Design and Implementation of XML based Workflow Management System

Myeongeun Oh[O] Seokhyun Yoon[O] Sanyong Han
Dept. of Computer Science and Engineering Chung-Ang Univ.

요 약

현대 사회의 급변하는 기업환경에 대처하기 위해서 기업들은 기존의 애플리케이션 사이에 중간 매개체인 미들웨어를 설치하여 애플리케이션 통합을 이룰 수 있는 EAI를 도입하고 있다. 그러나 현재의 EAI는 상이한 미들웨어를 사용하는 시스템 간의 통합을 위해서 Point-to-Point 방식의 코딩을 사용해야 한다는 문제점이 있다.

본 논문에서는 스테이트 차트와 XML 기술을 이용하여 다양한 미들웨어 시스템 상의 애플리케이션을 효율적으로 통합할 수 있는 미들웨어 독립적인 워크플로우 매니지먼트 시스템을 제안한다.

1. 서 론

최근 대부분의 기업들은 특정 작업을 위해서 하나의 새로운 애플리케이션을 개발하여 일괄 처리를 하기보다는 기존의 애플리케이션에 외부의 애플리케이션을 도입하여 그들을 유기적으로 결합하는 워크플로우 메니지먼트 시스템을 사용하고 있다. 또한 이러한 일련 작업을 위한 애플리케이션들의 통합뿐만 아니라, 기업 전체를 통해서 상호 연관된 모든 애플리케이션을 유기적으로 연동시켜 필요한 정보를 통합하고 관리, 사용할 수 있게 하는 EAI(Enterprise Application Integration)가 최근 큰 관심을 끌고 있다. 이러한 애플리케이션 통합은 재사용성, 유지보수성, 무결성, 안전성에 뛰어난 효과를 얻을 수 있다.

본 논문에서는 애플리케이션의 변화를 최소화하면서 애플리케이션을 통합, 관리할 수 있는 미들웨어 독립적인 워크플로우 매니지먼트 시스템을 제시하고자 한다. 본 논문에서 제시하고 있는 워크플로우 매니지먼트 시스템은 추가적인 애플리케이션 코드의 수정 없이 미들웨어 독립적으로 애플리케이션을 통합할 수 있도록 아답터를 사용하며, 통합된 애플리케이션 사이의 원활한 데이터 교환을 위해서 데이터 교환 표준으로 XML을 사용하였다. 애플리케이션 사이의 상호작용 및 원활한 데이터의 흐름을 제어하는 플로우 모델링을 위하여 스테이트 머신의 개념을 도입하고 이를 XML로 표현하였다.

본 논문의 구성은 다음과 같다. 우선 2장에서는 기반 연구로 EAI 및 EAI를 위한 아답터에 대한 설명과 함께 본 시스템의 개발에 기반이 된 스테이트 차트 개념을 설명하고, 3장 및 4장에서는 제시하는 시스템의 설계 및 구현에 대하여 설명하고, 마지막 5장에서는 결론 및 향후 연구 과제에 대해서 기술하였다.

특히 3, 4장의 경우 워크플로우 모델링을 위한 XML 포맷과 함께 닷넷 환경에서 C# 언어로 구현된 워크플로우 매니지먼트 시스템에 대한 자세한 설계 및 구현 사항이 포함되어 있다. 본 워크플로우 매니지먼트 시스템은 애플리케이션들 사이의 워크플로우를 제어하는 워크플로우 엔진과 이러한 워크플로우를 동적으로 모델링 할 수 있는 워크플로우 모델링 클라이언트로 구성되어 있다.

2. 기반 연구

2.1 EAI

EAI는 기업내부에 상이한 애플리케이션과 비즈니스 프로세스를 통합하는 IT 솔루션으로, 미들웨어를 이용해 비즈니스 로직을 중심으로 기업 내 각종 애플리케이션을 통합하는 과정이다.

상호 연관된 모든 애플리케이션을 유기적으로 연동하여 필요한 정보를 중앙 집중적으로 통합, 관리, 사용할 수 있는 미들웨어를 구축하는 것이 EAI의 역할이라 할 수 있다.

2.2 Statechart

Statechart는 David Harel이 반응시스템을 명세하기 위해 제한한 가시적 정형기법으로서[1][2], 기존 상태추이도(State-transition digaram)에 계층성(hierarchy), 병행성(concurrency), 동보통신(broadcast-communication)개념을 추가해 확장한 것으로서, 상태와 전이를 쉽게 클러스터화 하거나 구체화시킬 수 있는 정형기법이다.

시스템의 동적인 동작을 기술하기 위한 정형 명세 언어로 사용하기 위해 Harel의 statechart를 다음과 같이 보완한다. 각각의 상태 내에서는 트리거(trigger, 이벤트 혹은 조건)가 정의되고, 트리거가 발생하면 전이(transition)가 일어날 목적지 상태(target state)와 이때 해 주어야 할 액션(action)이 정의되며 현재 상태 내에서 실행해야할 행위(activity)를 현재 상태에 처음 들어올 때 한번 해주는 부분(entry action), 상태에 머무르는 동안 행하는 부분(do action), 그리고 상태를 빠져나갈 때 해주어야 하는 부분(exit action)으로 나누어 정의한다.[3]

3. 워크플로우 매니지먼트 시스템의 분석

3.1 시스템 분석

기본적으로 워크플로우 매니지먼트 시스템은 ① 기존 시스템 및 시스템에 포함되어 있는 서비스 정보를 모델링 할 수 있어야 하고, ② 워크플로우를 모델링 할 수 있는 기술 언어가 정의되어야 하며, ③ 대상 시스템의 서비스를 호출할 수 있는 프로토콜이 정의되어야 한다. 그리고 ④ 워크플로우의 모델링을 효율적으로 수행하기 위한 사용자 친화적 인터페이스를 제공해야 하고, ⑤ 워크플로우를 모델링한 기술 언어 자료를 해석하고 실행 할 수 있는 엔진이 제공되어야 하며, ⑥ 이 기종의 미들웨어 사이의 통신을 위해서 아답터를 제공해야 한다.

3.2 워크플로우 모델링

시스템 사이의 워크플로우를 모델링하기 위해서는 시스템이 가지고 있는 서비스의 목록과 이벤트를 비롯한 이들 서비스에 대한 속성 정보 등을 모델링 할 필요가 있다. 이러한 시스템의

서비스 정보를 모델링 하는 것이 정적 모델링이다. 다음으로 이러한 정적 모델링 과정에서 정의되는 시스템 정보를 바탕으로 시스템 사이의 프로세스의 흐름을 모델링 해야 하는데, 이러한 모델링이 동적 모델링이 된다. 즉, 이러한 정적 모델링과 동적 모델링이 함께 이루어져야 올바른 워크플로우 모델링이 가능하다. 본 논문에서 구현한 워크플로우 매니지먼트 시스템에서는 이러한 정적, 동적 모델링을 위한 그래픽 사용자 인터페이스 환경을 제공하고 있으며, 모델링 결과는 XML로 저장된다.

3.2.1 정적 모델링

```
- <systems>
  - <system>
      <name>A</name>
    - <services>
      - <service seqType="S">
          <name>ServiceA1</name>
        - <timeOuts>
            <timeOut>0</timeOut>
            <timeOut>100</timeOut>
            <timeOut>200</timeOut>
          </timeOuts>
        - <variables>
            <variable>a</variable>
            <variable>b</variable>
            <variable>c</variable>
          </variables>
        </service>
      - <service seqType="A">
          <name>ServiceA2</name>
          <timeOuts />
        - <variables>
            <variable>d</variable>
          </variables>
        </service>
      </services>
    </system>
  - <system>
      <name>B</name>
    - <services>
      - <service seqType="S">
          <name>ServiceB</name>
        - <timeOuts>
            <timeOut>100</timeOut>
          </timeOuts>
        - <variables>
            <variable>e</variable>
          </variables>
        </service>
      </services>
    </system>
  </systems>
```

[그림 2] 시스템 정보를 기술하는 정적 모델링

정적 모델링은 기존에 존재하는 애플리케이션에 대한 시스템 정보를 모델링한다. 모델링의 대상이 되는 시스템 정보에는 제공하는 서비스의 이름과 연결 타입, 응답 제한 시간 및 사용되는 변수 등의 정보가 포함된다. 연결 타입이란 서비스 즉, 시스템이 제공하는 메소드가 동기 혹은 비동기 호출을 수행하는지를 나타낸다. 응답 제한 시간은 동기 호출의 경우 시스템 대기 한도 시간을 의미하고, 변수는 시스템간 통신을 위해 사용되는 저장소를 의미한다.

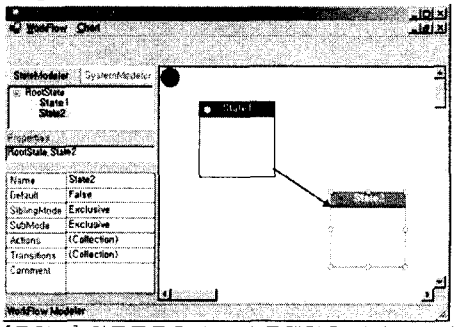
<표 1> 시스템의 서비스 정보

시스템	서비스	sync	timeout	variables
A	ServiceA1	S	0 100 200	a b c
	ServiceA2	A	-	d
B	ServiceB	S	100	e

<표 1>은 ServiceA1과 ServiceA2를 가진 시스템 A와 ServiceB를 가진 시스템 B에 대한 시스템의 정적인 정보이며, 이들 정보를 모델링한 XML 결과는 [그림 2]와 같다.

3.2.2 동적 모델링

동적 모델링은 기반 연구에서 소개한 David Harel의 스테이트 차트를 개선한 형태의 모델을 적용하여 정의한다. 메시지 혹은 이벤트 등에 의한 일련의 상태 변화를 스테이트로 기술하고, 각각의 스테이트에서 수행되어야 할 행위를 액션으로 정의함으로써 워크플로우의 동작을 모델링 할 수 있게 된다. 워크플로우의 동작은 스테이트 진입시 수행할 행위와 스테이트에 머무는 동안 실행할 행위, 스테이트를 빠져 나갈 때 수행할 행위를 각각 스테이트 별로 나누어 작성한다. 각각의 스테이트는 이벤트나 메시지와 같은 트리거에 의해서 상태 전이가 이루어지며, 각각의 스테이트마다 정의되어진 액션에 따라서 워크플로우의 동적인 행위가 처리되게 된다.



[그림 3] 워크플로우의 동적 모델링을 위한 사용자 인터페이스

본 시스템은 위의 동적인 스테이트를 모델링하기 위한 사용자 친화적인 그래픽 인터페이스를 제공하고 있다. [그림 3]은 동적 모델링을 위한 사용자 인터페이스의 모습으로, 시스템 A의 ServiceA2 이벤트가 발생하였을 경우 시스템 B의 ServiceB 메소드를 호출하는 워크플로우를 모델링하고 있다. 초기 상태인 State1에서 시스템 A의 ServiceA2 이벤트가 발생하면 State1에서 State2로 상태 전이가 발생하며 State2의 ENTRY 액션으로 지정된 시스템 B의 ServiceB가 실행된다. [그림 4]는 위크플로우의 동적 모델링을 수행한 결과로 생성되는 XML 문서이다.

3.3 이기종의 시스템 간의 통신

본 워크플로우 매니지먼트 시스템에서는 이기종 시스템 간의 통신의 문제점을 해결하기 위해 기본 플랫폼을 마이크로소프트의 닷넷 플랫폼을 바탕으로 설계하였다. 마이크로소프트 닷넷 플랫폼은 기존 COM과의 통신을 지원하며 기존의 다양한 미들웨어들의 드라이버를 COM으로 연결하여 마이크로소프트 닷넷과 직접 통신시키는 시나리오가 적용 가능하였다.

4. 워크플로우 매니지먼트 시스템의 설계 및 구현

본 워크플로우 매니지먼트 시스템은 마이크로소프트의 닷넷 플랫폼에서 C# 언어로 구현되었으며, XML과 스테이트 차트를 이용하여 설계하였고, 이기종 미들웨어 간의 메시징을 위해서 EAI 아답터를 적용하여 구현하였다.

제안하고 있는 워크플로우 매니지먼트 시스템은 3-계층(3-tier) 모델을 사용하고 있다. 클라이언트 계층에서는 사용자가 쉽게 워크플로우를 모델링하고, 워크플로우의 동작을 모니터링할 수 있는 사용자 친화적인 인터페이스를 제공한다. 그리고 데이터베이스 계층에서는 시스템의 정적인 정보와 사용자가 모델링한 동적인 워크플로우 정보를 저장하며, 서버 계층에서는 데이터베이스 계층과 클라이언트 계층과 상호 작용하며 워크플로

우를 관리하고 실행하는 워크플로우 엔진이 존재하게 된다.

```
<?xml version="1.0" encoding="utf-8"?>
<stateDef>
  <state guid="root" subMode="EXC" siblingMode="EXC" default="N">
    <name>RootState</name>
    <transitions></transitions>
    <actions></actions>
    <state guid="a" subMode="EXC" siblingMode="EXC" default="Y">
      <name>State1</name>
      <transitions>
        <transition>
          <nextState>State2</nextState>
          <triggerDef>
            <trigger relation ="OR">
              <trigger relation="NO">
                <system>A</system><service>ServcieA2</service>
              </trigger>
            </trigger>
          </triggerDef>
          <actions>
            <action when="TRANSITION" seqType="S" timeOut="0">
              <system>A</system><service>ServiceA1</service>
              <variable>b</variable>
            </action>
          </actions>
        </transition>
      </transitions>
      <actions></actions>
    </state>
    <state guid="b" subMode="EXC" siblingMode="EXC" default="N">
      <name>State2</name>
      <transitions></transitions>
      <actions>
        <action when="ENTRY" seqType="S" timeOut="0">
          <system>B</system><service>ServcieB</service>
          <variable>e</variable>
        </action>
      </actions>
    </state>
  </state>
</stateDef>
```

[그림 4] 워크플로우의 동적 모델링 결과

4.1 클라이언트 계층

본 워크플로우 매니지먼트 시스템은 효과적인 워크플로우의 정적, 동적인 모델링을 위하여 [그림 3]과 같은 사용자 친화적인 그래픽 인터페이스 환경을 제공한다. 모니터링 기능 및 아답터 정보 검색에 대한 내용들도 제공함으로써 더욱 사용자에게 편리한 환경을 제공한다.

4.2 데이터 베이스 계층

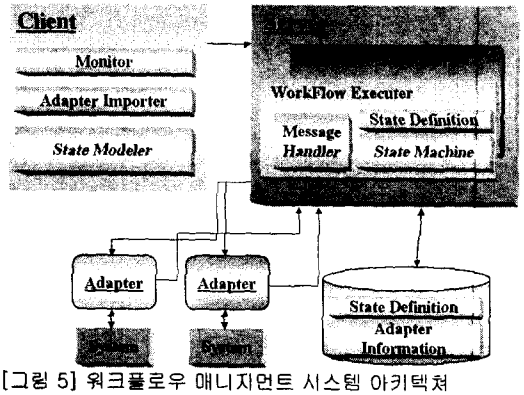
각각의 시스템들의 서비스 정보, 즉 워크플로우를 위한 정적인 모델링 정보를 저장하는 역할을 담당할 뿐만 아니라 클라이언트에서 작성된 워크플로우의 동적 모델링 정보를 저장한다. 즉, 워크플로우 매니지먼트 시스템의 전체적 운용을 위한 데이터들을 저장하게 되는 역할을 담당한다. 워크플로우 운용을 위한 데이터를 데이터 베이스 계층으로 분리하여 관리하기 때문에, 워크플로우의 변경 관리 및 버전관리를 용이하게 수행할 수 있다.

4.3 서버 계층

서버 계층에는 클라이언어트 계층에서 작성하게 되는 워크플로우의 동적인 모델링 정보와 데이터 베이스 계층에 저장되어 있는 시스템에 대한 정적인 모델링 정보를 바탕으로 워크플로우를 해석하고 실행하며 관리하는 워크플로우 엔진이 존재한다.

워크플로우 엔진은 워크플로우를 해석하고 실행하는 핵심 요소로, 워크플로우마다 프로세스의 형태로 생성된다. 워크플로우 엔진은 스테이트 정의 및 스테이트 머신으로 구성되어 있다. 메모리 상에 DOM 객체의 인스턴스로 관리되고 있는 워크플로우의 모델링 정보를 사용하여 스테이트 정의를 구성하게 되며, 스테이트 머신은 스테이트 정의에 따라 각 스테이트를 순회하면서 정의된 작업을 수행하게 된다. 그리고 스테이트 머신은 스테이트 정의에 존재하는 트리거 정보에 해당하는 이벤트나 메시지가 감지되면 다음 스테이트로 전이된다. 이러한 전이와 수행을 반복하면서 모든 워크플로우 작업을 완성하게 된다.

본 워크플로우 매니지먼트 시스템은 기존 시스템과의 상호 통신을 위하여 아답터를 사용하고 있으며, 워크플로우 엔진은 아답터와의 통신을 위해서 커넥션을 자동으로 생성하게 된다. 아답터는 기존에 운영중인 시스템과 워크플로우 메니저와의 중간 다리 역할 즉, 기존의 운영중인 시스템과 워크플로우 간의 통신을 이어주는 역할과 데이터의 변환 및 이기종간의 통신을 위한 드라이버 실행 코드의 호출 등을 담당하게 된다.



[그림 5] 워크플로우 매니지먼트 시스템 아키텍쳐

5. 결론 및 향후 연구 과제

본 논문에서는 데이터 통신을 위한 아답터와 워크플로우 매니지먼트 시스템을 개발함으로써 기존의 애플리케이션 통합하는데 쉬운 환경을 제공할 수 있도록 하였다. XML을 데이터 포맷으로 사용함으로써 이 기종 간의 데이터 교환을 가능하게 하였으며 미들웨어 간 통신을 위하여 아답터와 닷넷의 기술을 적용하였다.

본 워크플로우 매지니먼트 시스템은 애플리케이션 통합을 위해 기존의 애플리케이션의 수정을 최소화하고도 데이터 및 통신에서 통합을 가능하게 하며 이를 통해 대기업 등 기존의 기업들의 소프트웨어 유지 보수 비용을 상당수 절약할 수 있게 될 것이다.

본 워크플로우 매니지먼트 시스템의 워크플로우 엔진은 기존 애플리케이션에서 올라온 데이터를 런타임시에 동적으로 가공하지 못하는 문제점이 있다. 향후에는 아답터 레벨에서의 데이터 가공 뿐만이 아니라 스크립트 언어를 이용하여 런타임에 데이터를 가공함으로써 더욱 더 유연한 애플리케이션 통합 환경 제공을 연구하고자 한다. 그리고 애플리케이션 간 통신 트래픽량이 상당히 많을 경우 워크플로우 메니저에서 병목현상을 유발할 수 있으므로 효과적인 트래픽 분산과 함께 안정성을 확보할 수 있는 운용방법이 고려되어야 한다.

6. 참고 문헌

[1] D.Harel, "Statecharts: A Visual Formalism for Complex Systems." Science of Computer Programming, 8, pp. 231 - 274, 1987.

[2] D.Harel, A.Pnueli, J.P.Schmidt and R.Sherman, "On the formal semantics of statecharts", Proc. 2nd IEEE Sysposium on Logic in Computer Science, pp 54-64, 1987.

[3] 김철웅,한상용,최진영,이정아 " IP/Sim : Statecharts에 기반을 둔 가상 프로토타이핑 시뮬레이터 설계 및 구현" 한국정보처리학회 논문지 제7권 제3호, pp.891 - 900. 2000.