다중 시나리오를 지원하는 상황인지 워크플로우

양홍준* 최종선 조용윤 최재영 유재우
숭실대학교 컴퓨터학과
e-mail : {garamyunseul*, jschoi}@ss.ssu.ac.kr sslabyycho@hotmail.com {choi, cwyoo}@ssu.ac.kr

A Context-Aware Workflow Supporting Multiple Scenarios

Hongjun Yang Jongsun Choi Youngyun Cho Jaeyoung Choi Chaewoo Yoo
Dept. of Computing, Soongsil University

요 약

유비쿼터스 컴퓨팅 환경에서는 동적으로 발생하는 수많은 사용자나 환경에 대한 정보를 수집하여 적합한 서비스를 사용자에게 제공해야 한다. 이러한 유비쿼터스 환경을 워크플로우 형태로 기술할 수 있는 uWDL 을 이용하여 동적으로 변화하는 환경의 각각의 사용자에게 적합한 서비스를 제공하기엔 한계가 있다.

이를 해결하기 위한 방법으로 본 논문에서는 다중 시나리오를 지원하기 위한 방법인 멀티플로우와 서브플로우를 시나리오에 적용하여 해결 방안을 제시한다.

1. 서론

최근의 기업이 커짐으로 인해 비즈니스 프로세스는 더욱 복잡해지고 데이터와 서비스 영역(service domain)이 커짐에 따라 시스템 구축 및 유지보수에 많은 비용이 소모된다. 이러한 문제를 해결 하기 위한 방법으로 제시된 워크플로우 기술은 많은 연구를 통해 1990 년대 중반부터 현재까지 다양한 비즈니스 프로세스의 자동화와 효율성을 극대화했다.

워크플로우는 각 업무의 구성을 순차로 나타내어 누가 수행하고 어떻게 흐르는지 추적이 가능하다는 장점이 있기 때문에 비즈니스 프로세스에 적합하다. 현재 워크플로우의 개념은 웹 서비스 기반의 비즈니스 프로세스를 기술하기 위한 표준안인 WSBPEL(Web Service Business Process Execution Language)[1]나 ERP(Enterprise Resource Planning), BPM (Business Process Management)[2] 시스템들에서 비즈니스 프로세스를 구현하기 위하여 널리 사용되고 있다.

이를 기반으로 하는 상황-인지(Context-Aware) 워크플로우는 워크플로우의 장점을 이용하여 유비쿼터스 환경에 적용시키기 최근에 연구되고 있는 기술이다. 그러나, 전통적인 비스니스 프로세스의 자동화 모델인 워크플로우를 동적인 특성의 유비쿼터스 환경에 적용하기 위해서는 다양하고 다중적인 서비스 플로우에 대한 요구를 워크플로우에 표현할 수 있는 시나리오 기술이 요구된다. 즉, 유비쿼터스 환경에서의 워크플로우는 동적으로 생성되고 소멸되는 다양한 워크플로우가 존재하게 된다. 이때 각 워크플로우는 일부나 전체가 동시성을 가지게 될 수도 있고, 한 워크플로우에 다른 워크플로우가 포함되어 종속성을 가지게 될 수도 있다. 따라서, 이러한 동적인 워크플로우의 특성을 전체적인 흐름속에 유기적으로 표현할 수 있는 워크플로우 시나리오 기술은 상황인지 워크플로우 서비스 기발 기술로써 관련 응용프로그램 개발에 활용도가 클 것으로 기대된다.

본 논문에서는 관련 연구로써 기존의 웹 서비스 기반 워크플로우 언어인 WSBPEL 와 정적인 특성의 워크플로우를 유비쿼터스 환경에 적용하기 위한 시나리오 기술 방법인 uWDL(Ubiquitous Web service Definition Language)[3,4]에 대해 소개한다. 본론에서는 단일 워크플로우만을 서비스 시나리오로 표현할 수 있었던 기존의 상황인지 워크플로우 시나리오 기술의 문제점을 해결하기 위한 새로운 시나리오 기술에 대해 설명한다. 설명한다. 또한, 멀티플로우(MultiFlow)와 서브플로우(SubFlow)로 표현되는 시나리오 표현 실험을 통해 제안한 상황인지 워크플로우 시나리오 기술의 유용성을 보인다. 마지막으로 연구 결과와 향후 연구 방향에 대해 설명한다.

2. 관련연구

2.1 웹 서비스 기반 워크플로우 시나리오 기술

유비쿼터스 환경에서의 서비스는 이질적인 컴퓨팅 개체들 사이에서 오류 없이 동작할 수 있어야 한다. 웹 서비스는 이질적인 컴퓨팅 기기들 간의 유기적인 서비스 흐름을 위해 검증된 다양하고 유용한 기술을 제공한다. WSBPEL 는 IBM 과 MS 그리고 BEA 가 공동으로 작업하여 만든 웹 서비스 기반의 비즈니스 프로세스를 기술하는 XML 기반 표준 워크플로우 언어이다.

WSBPEL 은 IBM 의 WSFL 과 MS 의 XLANG 두 가지 언어의 장점을 합병하여 웹 서비스들을 연결하고 조합해 모든 종류의 비즈니스 프로세스를 구현할 수

있다. WSBPEL 는 서비스의 자동화를 위한 실행 조건으로 정적인 메시지를 기반으로 한다. 따라서, 유비쿼터스 환경에서의 동적으로 발생하는 사용자 상황정보를 서비스의 실행 조건으로 기술해야 하는 상황인지 워크플로우 시나리오 기술로써 그 활용도가 충분하지 않다.

2.2 유비쿼터스 환경 기반 워크플로우 시나리오 기술

uWDL(Ubiquitous Web service Definition Language)은 유비쿼터스 환경(Ubiquitous Enviroments)에서 사용자에게 제공할 웹 서비스와 이들의 실행조건으로 사용자의 상황정보를 미리 정의된 워크플로우(Workflow) 시나리오로 표현할 수 있는 XML 기반 워크플로우 언어이다. 이를 위해, uWDL 은 유비쿼터스 환경에서 발생하는 모든 컨텍스트, 프로파일 및 이벤트 정보에 대해 워크플로우 시나리오에 표현할 수 있는 노드 엘리먼트를 포함한다. 상황인지 워크플로우 개발자는 uWDL 을 이용하여 서비스들 간의 흐름을 온톨리지 형태의 규칙 기반(Rule-based) 컨텍스트 기술 방식의 XML 시나리오로 표현할 수 있다.

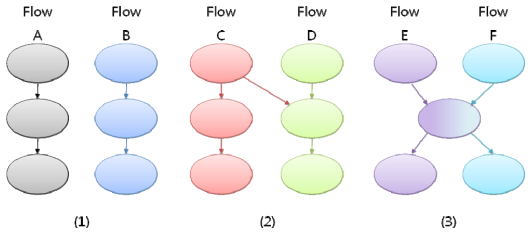
그러나, uWDL 은 서비스 실행조건으로써 사용자 상황정보를 단일 워크플로우로만 표현 가능하며, 동적으로 변화되는 상황정보를 다중의 워크플로우에 적용하거나 기존 워크플로우의 형태를 동적으로 변경시킬 수 있는 기능을 언어수준으로 제공하지 않는다. 따라서, 기존 uWDL 을 이용하는 상황인지 서비스 개발자는 실제 유비쿼터스 환경에 적응하는 응용프로그램 개발에 제한을 받는다.

3. 본론

uWDL 과 같은 기존 유비쿼터스 환경을 위한 워크플로우 언어는 유비쿼터스 환경의 동적인 특성을 상황인지 워크플로우 시나리오에 충분히 적용하기에는 어려움이 있다. 본 논문에서는 기존 uWDL 이 단일 워크플로우만을 지원하는 제약점을 해결하고, 다중 워크플로우를 지원할 수 있는 워크플로우 시나리오 기술을 제안한다. 이를 위해서, 본 논문은 uWDL 의 기본적인 언어 스키마를 사용하여 멀티플로우와 서브플로우에 대한 실제 다중 워크플로우 시나리오를 설계한다. 다음은 본 논문에서 제안하는 다중 워크플로우 시나리오를 위해 사용되는 멀티플로우와 서브플로우의 정의이다.

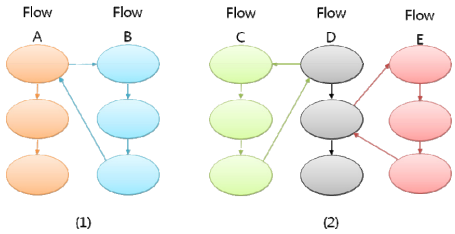
1. 두 개 이상의 워크플로우가 공존하는 경우를 멀티플로우라고 정의한다.
2. 워크플로우 A 안에 워크플로우 B 를 포함하고 있는 경우에 플로우 B 를 플로우 A 의 서브플로우라고 정의한다.

다음 (그림 1)은 멀티플로우 예를 그림으로 나타낸다. 멀티플로우를 나타내는 (그림 1-1)은 서로 간에 영향을 미치지는 않지만 동시성을 가지는 흐름을 보여준다. (그림 1-2)는 플로우 C 의 첫번째 노드의 결과에 따라 플로우 D 의 두번째 노드에 영향을 준다. (그림 1-3)은 플로우 E 와 F 의 두번째 노드가 하나로 합쳐져 있는데, 이는 동일한 서비스를 E 와 F 에서 같은 시점에 제공함을 나타낸다.



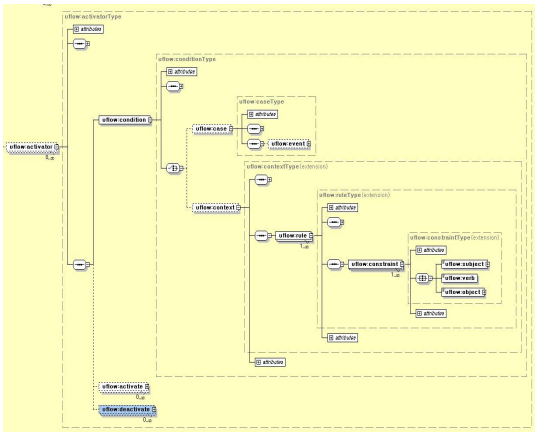
(그림 1) 멀티플로우의 예

다음 (그림 2)는 서브플로우 예를 그림으로 나타낸다. (그림 2-1)은 플로우 B 는 플로우 A 의 서브플로우가 되며, (그림 2-2)에서는 플로우 C 와 E 둘다 플로우 D 의 서브플로우가 된다.



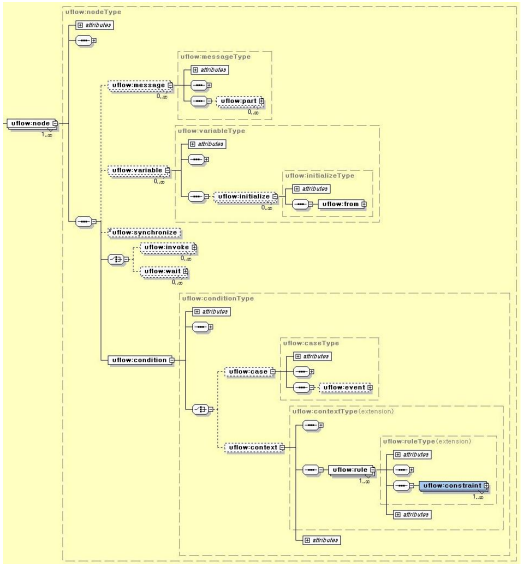
(그림 2) 서브플로우의 예

본 논문에서는 멀티플로우와 서브플로우를 지원하는 다중 워크플로우 시나리오를 위해 기존 uWDL 스키마의 엘리먼트 <activator>를 (그림 3)에서와 같이 확장한다. 다음 (그림 3)은 uWDL 에서 플로우를 활성화하는 부분인 <activator>의 스키마를 보여준다.



(그림 3) 확장된 uWDL <activator> 엘리먼트

하나의 플로우가 실행되기 위해서는 <activator> 엘리먼트 내에 정의된 컨텍스트나 이벤트 또는 조건 비교를 통해 활성화된다. 이 때, 여러 개의 <activator>에서 각각 하나 이상의 플로우를 활성화 시킬 수 있다. 즉, 여러 개의 <activator>를 하나의 워크플로우 안에서 개별적인 서비스 노드로 활성화 시키면, 독립적으로 실행되는 다중 워크플로우를 얻을 수 있다. 또한, 서브플로우를 통한 다중 워크플로우 시나리오는 (그림 4)의 uWDL 스키마에서 <node>부분의 확장을 통해 얻을 수 있다. 다음 (그림 4)는 uWDL 에서 워크플로우 액티비티를 나타내는 부분인 <node> 엘리먼트의 스키마를 보여준다.



(그림 4) 확장된 uWDL <node> 엘리먼트

서브플로우는 워크플로우의 한 노드에서 다른 워크플로우의 <activator>를 활성화하는 방식을 취한다.

4. 실험 및 평가

본 논문에서 제안하는 확장된 uWDL 엘리먼트에 대해 다음과 같은 예제 시나리오를 이용하는 상황인지 다중 워크플로우 시나리오의 설계를 실험하였다.

시나리오 1

멀티플로우를 포함하는 uWDL 시나리오를 예로 보이기 위해 서비스 도메인을 회사 사무실로 정하고 직원 A 와 B 가 각 해당 사무실에 도착하는 경우 A 와 B 를 위한 각각의 워크플로우를 활성화 시킨다.

“ 인사팀 A 씨는 아침 9 시에 출근하여 302 호 인사팀 사무실로 입장한다. A 씨 자리의 RFID 센서는 A 씨의 ID 카드의 신상 정보를 읽고 현재 시간이 9 시 이고 A 씨가 자리에 앉았다는 정보가 확인되면, 해당 워크플로우를 활성화시킨다.”

“기획팀 B 씨는 아침 9 시 30 분에 출근하여 210 호 기획팀 사무실로 입장한다. B 씨 자리의 RFID 센서는 B 씨의 ID 카드의 신상 정보를 읽고 현재 시간이 9 시 30 분 이고 B 씨가 자리에 앉았다는 정보가 확인되면, 해당 워크플로우를 활성화시킨다.” <표 1>은 (그림 1-1)의 멀티플로우를 시나리오 1 에 따라 uWDL 을 이용해 기술한 것이다.

<표 1> 시나리오 1 을 적용한 멀티플로우의 예

```
<activator name="Activator1">
 <condition expression="c1 and c2">
  <context>
   <rule>
    <constraint name="c1">
     <subject type="militaryTime">currentTime1</subject>
     <verb>is </verb>
     <object type="militaryTime">0900</object>
    </constraint>
    <constraint name="c2">
     <subject type="personType">personID1</subject>
     <verb>located</verb>
     <object type="roomType">302</object>
    </constraint>
   </rule>
  </context>
 </condition>
<activate flow="FlowA"/>
</activator>
<activator name="Activator2">
 <condition expression="c1 and c2">
  <context>
   <rule>
    <constraint name="c1">
     <subject type="militaryTime">currentTime2</subject>
     <verb>is </verb>
     <object type="militaryTime">0930</object>
    </constraint>
    <constraint name="c2">
     <subject type="personType">personID2</subject>
     <verb>located</verb>
     <object type="roomType">210</object>
    </constraint>
   </rule>
  </context>
 </condition>
<activate flow="FlowB"/>
</activator>
<flow name="FlowA">
    //중략//
</flow>
<flow name="FlowB">
    //중략//
</flow>
```

<표 1>의 Activator1 은 명시된 컨텍스트를 엔진에서 비교한 값이 참일 경우 FlowA 를 활성화 시킨다. Activator2 역시 컨텍스트의 비교 값이 참일 경우 FlowB 를 활성화시키게 되어 총 2 개의 워크플로우가 활성화되어 멀티플로우가 된다.

시나리오 2

서브플로우를 포함하는 시나리오의 예는 다음과 같다. 기획팀 B 씨는 오후 2 시에 회의실에서 발표가 있다. 오후 1 시 50 분에 발표 준비를 위해 B 씨가 미리 입장하면 회의 준비를 지원해주는 서비스가 실행된다.

"기획팀 B 씨는 자신의 자리에서 발표 자료를 서버로 전송한다. B 씨가 오후 2 시의 발표를 위해 501 호 회의실에 10 분 전 도착하게 되면, 서브플로우에 정의된 회의 준비 서비스가 지원된다." <표 2>는 (그림 2-1)의 서브플로우를 시나리오 2 에 따라 uWDL 을 이용해 기술한 것이다.

<표 2> 시나리오 2 를 적용한 서브플로우의 예

```
<activator name="PrepareMeetingRoomActivator">
 <condition expression="c1 and c2 and c3 not c4">
  <context>
   <rule>
    <constraint name="c1">
     <subject type="personType">personID2</subject>
     <verb>located</verb>
     <object type="roomType">501</object>
    </constraint>
    <constraint name="c2">
     <subject type="roomType">501</subject>
     <verb> reserved</verb>
     <object type="militaryTime">1400</object>
    </constraint>
   </rule>
  </context>
 </condition>
 <activate flow="CheckMeetingRoomFlow"/>
</activator>
<flow name="PlanB">
 <node name="Meeting">
  <condition expression="c1 and c2 and c3">
   <context>
    <rule>
     <constraint name="c1">
      <subject type="militaryTime">currentTime</subject>
      <verb>is </verb>
      <object type="militaryTime">1350</object>
     </constraint>
     <constraint name="c2">
      <subject type="personType">personID2</subject>
      <verb>located</verb>
      <object type="roomType">501</object>
     </constraint>
     <constraint name="c3">
      <subject type="roomType">501</subject>
      <verb>reserved</verb>
      <object type="militaryTime">1100</object>
     </constraint>
    </rule>
   </context>
  </condition>
 <invoke activator="PrepareMeetingRoomActivator"/>
 <wait serviceProvider="Company" portType="PlayMusicPT"
operation="PlayMusic" output="musicState"/>
 </node>
 //중략//
</flow>
<flow name=" CheckMeetingRoomFlow">
 //중략//
</flow>
```

플로우 PlanB 의 Meeting 노드에서 다른 워크플로우의 <activator>를 <invoke> 엘리먼트를 사용해 활성화시킨다. 그 후 <wait> 엘리먼트를 사용해 서브플로우에서 호출한 웹서비스 PlayMusic 의 반환 값이 되돌아 올 때까지 기다렸다가 다음 노드로 넘어가게 된다. 이처럼 한 워크플로우 내에서 다른 워크플로우를 호출하여 사용하여 서브플로우를 표현할 수 있다.

5. 결론

본 논문에서는 동적인 특성을 가지는 유비쿼터스 환경에서의 다양한 워크플로우의 동시성과 종속성을 표현할 수 있는 다중 워크플로우 시나리오 표현 기술을 제시하고 시나리오를 통하여 증명했다. 이를 위해 본 논문에서는 기존 uWDL 의 스키마를 확장하였다. 제안하는 시나리오 표현 기술은 실제 유비쿼터스 환경에서 발생하는 동적인 사용자 상황정보를 워크플로우 시나리오 문서에 표현할 수 있는 방법을 제공함으로써, 상황인지 워크플로우 응용프로그램 개발에 크게 기여할 것으로 기대된다. 향후에는 더욱 쉽고 빠르게 다중 워크플로우 시나리오 문서를 생성하고 시뮬레이팅 할 수 있는 그래픽 기반의 시나리오 설계 개발 도구에 대한 연구를 진행할 것이다.

참고문헌

[1] OASIS, WSBPEL v2.0(11 April 2007), http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

[2] OMG, BPM, http://www.bpmi.org

[3] 최종선, 조용윤, 최재영, 동적인 사용자 서비스 요구를 지원하는 상황인지 워크플로우 시스템, 시스템 및 이론 제 34 권 제 5,6 호 pp. 214~227, 2007. 6

[4] Kyuongho Shin, "AWorkflow Language for Context-Aware Services," mue,pp.1227-1232, 2007 International Conference on Multimedia and Ubiquitous Engineering (MUE'07), 2007.