다양한 비즈니스 프로세스 언어를 지원하는 적응적인 프로세스 설계 모델 개발*

조영현, 정문영, 탁경현, 손진현
한양대학교 컴퓨터 공학과
{mhjo, myjeong, khtak, jhson}@cse.hanyang.ac.kr

Developing an Adaptive Process Modeling Mechanism for Variable Business Process Languages

Myung Hyun Jo, Jung, Moon-young, Kyung-Hyun Tak, Jin Hyun Son
Department of Computer Science and Engineering, Hanyang University

요 약

최근 비즈니스 프로세스 통합에 관련된 다양한 연구를 통해, 수많은 비즈니스 프로세스 언어 및 표기법들이 개발되고 있다. 표준화되지 않은 다양한 비즈니스 프로세스 언어들은 비즈니스 프로세스를 자동화하려는 기업들뿐만 아니라, 비즈니스에 참여하는 일반 사용자에게도 혼동을 가져온다. 본 논문은 다양하고 표준화되지 않은 비즈니스 프로세스 언어들을 모두 디자인할 수 있기 위한 적응적인 모델을 제안한다. Jena와 JGraph와 같은 기존 모델의 장점들을 조합하고 단점을 보완하여 개발되었다. 본 논문에서 제안한 모델은 파이프-필터 아키텍쳐를 이용해 비즈니스 프로세스의 데이터를 단계적으로 추출한다. 그리고 추출된 데이터는 다양한 환경에 적응하기 위해, 해쉬 또는 링크드 리스트의 자료 구조에 저장되어 관리된다. 마지막으로, 저장된 데이터들은 사용자의 요구에 따라 실행언어로 변환되거나 다시 GUI에 보여 진다.

키워드: 모델링, 비즈니스 프로세스, BPMN, XPDL

1. 서 론

최근 EAI와 워크플로우등의 기존 비즈니스 프로세스 자동화 기술이 한계점을 나타내면서, Microsoft, IBM, BPMI, W3C와 같은 수많은 기관들은 기존 기술을 포함하면서 한계점을 극복하기 위한 비즈니스 프로세스 통합에 관한 연구를 활발히 진행하고 있다. 기존 자동화 기술의 문제점은 서로 다른 기술로 비즈니스 프로세스를 자동화하던 기업들 간의 비즈니스 프로세스 통합이 어렵다는 것이다. 서로 다른 자동화 기술을 통합하기 위해서는 내부 프로세스의 정보를 공개하거나, 내부 프로세스의 일부를 변경할 필요가 있다. 이것은 비즈니스를 제공하는 업체뿐만 아니라 비즈니스에 참여하는 고객, 파트너, 공급자 모두에게 경제적 손실을 가져다준다. 그래서 비즈니스 통합의 표준화에 관한 연구가 필요하며, 크게 두 분야로 나뉘어 진행되고 있다.

첫째, 비즈니스 프로세스 실행 언어에 대한 연구가 웹 서비스와 연동되어 진행되고 있다. 웹 서비스는 인터페이스를 통해 외부 프로세스에 쉽게 접근 할 수 있기 때문에, B2C뿐만 아니라 B2B까지 비즈니스 프로세스를 자동화할 수 있다. 대표적인 실행 언어로 WfMC에서 개발한 XPDL (XML Process Definition Language), IBM, BEA, Microsoft에서 공동 개발한 BPEL4WS (Business Process Execution Language for Web Service), BPMI에서 개발한 BPML (Business Process Modeling Language)을 예로들 수 있다.

둘째, 비즈니스 프로세스 실행언어는 비즈니스 프로세스를 사용하거나 디자인하는 사람을 위한 것이 아니라 프로세스를 이해하여 실행시키는 시스템을 위한 것이다. 그래서 사람이 쉽게 이해할 수 있는 비즈니스 프로세스 모델링에 대한 연구가 진행되고 있다. BPMN (Business Process Modeling Notation) [2], UML 액티비티 다이어그램, UML EDOC 비즈니스 프로세스, IDEF, ebXML BPSS, ADF (Activity-Decision Flow) 다이어그램, RosettaNet, LOVeM, EPCs (Event Process Chains) 등을 예로 들 수 있다.

두 연구 분야는 비즈니스 프로세스를 통합하는 데 공통된 목적을 갖고 동일한 의미의 비즈니스를 표현하려 한다. 하지만, 두 분야가 실제 구현에서 적용되는 범위는 확연히 구분된다. 전자는 서버 시스템에서, 후자는 GUI(Graphic User Interface)에 적용된다. 본 논문은 BPMN또는 BPEL4WS과 같은 다양한 비즈니스 프로세스 모델링 언어에 적응적으로 시스템이 대응할 수 있는 모델링 기법을 제안한다. 그림 1은 본 논문에서 제안한 모델링이 고려해야할 요소를 보여주며, GUI를 위한 Showing, 서버 시스템의 실행 언어를 위한 Mapping, 그리고 Storing으로 나뉜다. 특히, Showing과 Mapping은 비즈니스의 의미를 표현하는 부분으로 비즈니스 프로세스 모델에 능동적인 정보를 제공한다.

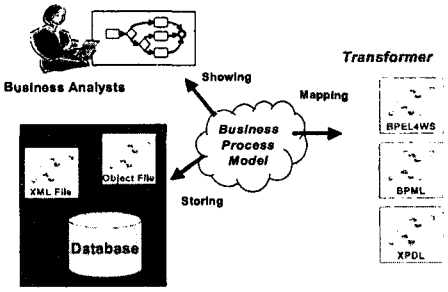

그림 1. 적응적인 비즈니스 프로세스 모델의 관련 요소

본 논문의 구성은 다음과 같다. 제 2장에서는 관련 연구로써 기존 모델링에 대한 연구를 분석한다. 제 3장에서는 다양한 비즈니스 프로세스 모델링 언어에 대한 적응적인 모델의 설계를 기술한다. 제 4장에서는 모델링 기법의 구현 및 결과를 BPMN과 XPDL의 예제를 통해 설명하고, 5장에서는 결론을 맺는다.

2. 관련 연구

2.1 JGraph

[3]는 그래프(Graph)를 시각화하기 위한 모델을 제공한다. JGraph는 Roots, Attribute 그리고 ConnectionSet로 구성되었

* 본 연구는 대학 IT연구 센터 육성·지원 사업의 연구 결과로 수행되었음
* 본 연구는 한국 과학재단 목적 기초 연구 (R08-2003-000-10464-0) 지원으로 수행되었음

으며, 해쉬의 자료 구조를 갖기 때문에 매우 적은 용량의 footprint를 가진다. 또, 그래프의 삭제, 복사, 붙이기 등의 다양한 편집 기능들을 제공하는 장점을 가지고 있다. 즉, JGraph는 그림 1의 Showing부분에서 상당히 강력한 기능들을 제공한다. 하지만, 그림 1의 Mapping 부분을 위한 기능들은 부족하다. 만일 각 객체를 순회하려면, Roots에 포함된 Port나 Edge등의 불필요한 요소들도 해석을 해야 한다. 즉, 비즈니스 환경에는 최적화되지 않아서 그림 1의 Mapping 부분에는 부적합하다.

2.2 Jena

[4]는 트리플 형식의 그래프 구조를 갖고 있는 시맨틱 웹 언어 RDF/OWL을 기반으로 모델을 구성하였다. 트리플은 세 개의 노드로 서브젝트(Subject), 프리디케이트(Predicate), 오브젝트(Object)로 나누어지는데, 각 노드가 자신이 속해있는 모든 트리플을 알 수 있기 때문에 노드를 중심으로 순회할 수 있는 특징이 있다. 이러한 특징은 XPDL과 같이 실행을 위한 언어에서는 다음 실행할 노드로 바로 접근할 수 있다는 점에서 장점이 될 수 있다. 하지만, 어떤 객체를 그릴 때마다 순차 검색을 한다면, 상당한 시간적 불편함을 갖게 될 것이기 때문에 그림 1의 Showing 부분에서는 부적합하다.

본 논문은 다양한 비즈니스 환경에 적응할 수 있는 모델을 구성하기 위해 위 두 모델의 장점을 적절히 재구성하였다.

3. 다양한 비즈니스 프로세스 언어에 대한 적응적 모델 설계

비즈니스 프로세스를 자동화하기 위해서는 비즈니스 프로세스 관리 시스템(Business Process Management System)이 필요하다. BPMS는 프로세스 디자인을 위한 모듈, 비즈니스 데이터를 저장할 모듈, 모든 비즈니스 프로세스의 상호 작용을 조율하는 모듈, 사용자와 상호작용을 위한 모듈, 외부 컴포넌트와 연결을 위한 모듈 등으로 구성될 수 있다. 본 논문은 비즈니스 프로세스 관리 시스템의 다양한 모듈 중 비즈니스 분석가와 사용자가 간/직접적으로 영향을 끼치는 비즈니스 프로세스 디자이너 모듈에 초점을 맞추었다.

비즈니스 프로세스 디자이너는 다양한 기능을 가지고 있지만, 가장 중요한 것은 비즈니스 분석가가 쉽고 빠르게 프로세스를 디자인할 수 있어야 하는 것이다. 또, 다양한 비즈니스 프로세스 언어 및 표기법에 적응적으로 대처할 수 있는 기술이 있어야 한다. 현재 비즈니스 프로세스 언어 및 표기법은 표준화된 기술이 없기 때문에, 통합된 방법을 통해 디자인된 비즈니스 프로세스가 다양한 언어들에 적용될 수 있어야 한다.

본 논문은 비즈니스 프로세스 디자이너가 갖추어야 할 특징 중 가장 중요한 두 특징을 적용시킬 수 있도록 그림 2와 같은 아키텍쳐를 설계 하였다. 그림 2는 그림 1의 중앙에 위치한 비즈니스 프로세스 모델의 상세 설계를 보여준다. 그림 2의 사용자 인터페이스(GUI)와 변환기(Transformer)는 적응적 설계를 위한 외부 모듈이다. 적응적 비즈니스 프로세스 모델은 다음과 같은 특징을 고려해야 한다.

- 적응성(Adaptability): 어떤 비즈니스 프로세스 언어 및 표기법도 표현할 수 있어야한다.
- 일관성(Consistency): 동일한 비즈니스 의미에 대해 동일하게 보여주고(Showing), 동일하게 변환(Transforming)될 수 있어야 한다.

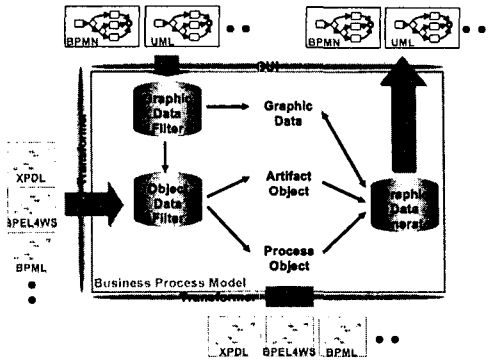

그림 2. 적응적인 비즈니스 프로세스 모델

적응적 비즈니스 프로세스 모델은 6 개의 컴포넌트로 구성되었으며, 각 컴포넌트들은 위에서 정의한 특징들을 내포하고 있다. 먼저 데이터 수집의 과정은 사용자 인터페이스(GUI)를 통할 경우와 변환기(Transformer)를 통할 경우 두 가지로 나뉘며, 구조는 파이프-필터 아키텍쳐를 이용하였다. 첫째, 그래픽 표기법의 위치 정보, 색깔, 크기, 글씨체와 같은 그래픽 데이터들이 사용자 인터페이스로부터 추출 되 야 만 한다. 그래픽 데이터는 그래픽 데이터 필터(Graphic Data Filter)를 통해 수집된다. 사실 이와 같은 데이터는 비즈니스 프로세스 자동화에는 어떤 영향도 끼치지 않는 데이터들이다. 하지만, 사용자에게 일관된 인터페이스를 유지하기 위해서 꼭 필요한 데이터들이다. 그리고 추출되고 남은 데이터는 객체 데이터 필터로 보내진다. 두 번째, 변환기로부터 수집된 데이터와 그래픽 데이터 필터에서 전해지는 데이터는 비즈니스 객체를 표현하는 데이터를 포함하고 있다. 비즈니스 객체 데이터는 객체 데이터 필터(Object Data Filter)를 통해, 아티팩트(Artifact) 객체와 프로세스(Process) 객체로 나뉘어 저장된다. 위의 두 컴포넌트는 외부 모듈로부터 전해지는 데이터를 파이프-필터 아키텍쳐를 통해 데이터를 그룹 핑하는 컴포넌트이다.

위에서 필터링 된 데이터들의 자료 구조는 다음과 같다. 첫째, 비즈니스 분석가가 쉽고 빠르게 비즈니스 프로세스를 디자인하기 위한 그래픽 데이터이다. 그래픽 데이터는 실제 비즈니스 객체(아티팩트, 프로세스)와 일대일 관계를 갖고 구성된다. 그리고 사용자와 직접적으로 연결되어 있기 때문에, 그래픽 데이터를 추출하는 데이터 검색 시간이 상당히 고려 되 야 할 사항이다. 그래서 본 논문은 그래픽 데이터의 빠른 검색을 위해 해쉬(Hash) 데이터 구조를 이용하였다. 그래픽 데이터 내용은 표1과 같고, 2장에서 기술한 JGraph의 장점을 수용하였다.[3]

표 1. 그래픽 데이터 구조

Key	Value
Business Object ID	Attribute(Location, Size, Color, Font)

둘째, 비즈니스 객체에 대한 데이터는 그림 3과 같은 계층 구조를 갖고 있다. 그림 3처럼 비즈니스 객체는 다시 두 부분

으로 나뉘는데, 비즈니스 프로세스 실행에 관련된 프로세스 객체와 비즈니스 프로세스 실행과는 관계없는 아티팩트 객체이다. 프로세스 객체는 액티비티(Activity) 또는 트랜지션(Transition)[1]과 같은 비즈니스 프로세스 실행에 직접적으로 관련이 있는 요소들을 지시한다. 즉, 실행 언어에 포함되는 요소들만을 가리킨다. 여기서 실행 언어들은 순차 프로세싱을 할 수 있도록 언어를 구성해야 하기 때문에, 프로세스 객체는 다중 연결 리스트(Multiple Linked List) 구조를 가져야 한다. 하나의 액티비티에 여러 개의 트랜지션을 가질 수 있는 특징은 하나의 연결 고리만을 갖는 연결 리스트(Single Linked List)의 자료 구조를 변경하게 했다. 프로세스 객체를 연결 리스트 구조로 선택한 가장 큰 이유는 비즈니스 프로세스 실행언어마다 구조가 모두 다르기 때문이다. 예를 들어, XPDL은 그래프(Graph) 구조를 갖고 있지만, BPEL4WS와 BPML은 블록 스터럭쳐(Block Structure)의 구조를 갖고 있다.[5] 이러한 구조를 일관성 있게 변경하기 위해서는 변경하기 쉬운 쪽을 기본 구조로 선택해야 한다. 그래프 구조는 블록 스트럭쳐보다 표현력이 방대하기 때문에, 블록 스트럭쳐에서 그래프 구조로 변경하는 것은 쉽다. 그래서 프로세스 객체의 구조는 그래프 구조가 아닌, 블록 스트럭쳐 구조인 다중 연결 리스트 구조를 적용하였다. 이것은 제 2장에서 기술한 Jena의 장점을 수용하였다.[4]

아티팩트 객체는 주석(Annotation)[2]처럼 비즈니스 실행과는 관련 없는 데이터를 지시한다. 아티팩트 객체는 앞서 기술한 그래픽 데이터처럼 사용자 인터페이스를 일관성 있게 유지하기 위해, 제거되어서는 안 되는 데이터이다. 그래서 해쉬 구조를 갖고 GUI에 보여 진다.

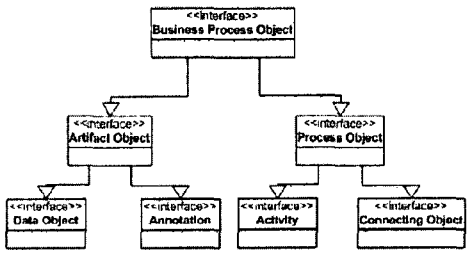

그림 3. 비즈니스 객체의 계층 관계

위와 같이 구성된 데이터들은 GUI에 표현되기 위해서 그래픽 데이터 생성자(Graphic Data Generator)에 이용되거나, 실행 언어를 위해서 변환기에 이용된다. 변환기로부터 입력된 비즈니스 객체 데이터는 그래픽 데이터를 갖고 있지 않기 때문에, 그래픽 데이터 생성자를 이용해 자동적으로 그래픽 데이터를 제공해야 한다. 기존에 그래픽 데이터가 저장되어 있으면, 그래픽 데이터 해쉬 테이블을 검색하여 데이터를 추출한다.

그림 4. CreditCheck Process

4. 구현 및 결과

본 장은 3장에서 기술한 적응적 모델의 설계를 BPMN과 XPDL을 이용해 구현한다. 그림 4는 XPDL 명세서[1]에 기술된 예제를 BPMN[2]을 이용해 표현한 것이다.

그림 4의 각 비즈니스 객체의 ID는, 해쉬 테이블의 키로써 적용된다. 표 2는 그래픽 데이터 필터를 통해 필터 링 된 그래픽 데이터들이 해쉬 테이블에 저장된 상태를 보여준다. 지면상 일부만을 보여준다.

표 2. 그래픽 데이터 필터를 통해 추출된 그래픽 데이터

Key	Value
Start	(0,10), (10,10), Black, BOLD
Set Credit Info	(10,10), (30,20), Red, BOLD
...	...

그 후, 객체 데이터 필터를 통해 프로세스 객체와 아티팩트 객체를 추출한다. 그림 4의 예제는 아티팩트 데이터를 포함하고 있지 않기 때문에, 객체 데이터 필터를 통해 프로세스 객체만을 데이터로 추출하게 된다. 표3은 추출된 프로세스 객체를 순차적으로 나열한 것이다. 각 객체는 내 부적으로 자신의 ID와 연결된 프로세스 객체의 참조한 데이터를 저장하고 있다.

표 3. 객체 데이터 필터를 통해 추출된 프로세스 객체

Process	ID	Next Process (Ref.)
1	Start Route	Flow 1
2	Flow1	Set Credit Info
...	...	...

마지막으로, 구성된 데이터들은 그래픽 데이터 생성자를 통해 다시 GUI에 보여 지거나, 변환기를 통해 그림 5처럼 XPDL 문서로 변환될 수 있다.

```
<Activities>
    <Activity Id="Set Credit Info" Name="Set Credit Info">
        <Route/>
    </Activity>
    <Activity Id="CreditCheck" Name="CreditCheck">
        <Implementation>
            <Tool Id="tmp" Type="APPLICATION">
                <ActualParameters/>
            </Tool>
        </Implementation>
    </Activity>
    <Activity Id="Start" Name="Start">
        <Route/>
    </Activity>
    ...
</Activities>
<Transitions>
    <Transition Id="Flow1" Name="Flow1"/>
    <Transition Id="Flow2" Name="Flow2"/>
    ...
</Transitions>
```

그림 5. 그림 4로부터 변환한 XPDL 문서의 일부

5. 결론

본 논문에서 제안한 적응적인 비즈니스 프로세스 모델은 다양한 비즈니스 프로세스 언어 및 표기법을 모두 표현하기 위해, 2장 모델들의 장점들만을 조합하여 구성하였다. 또, 기존 비즈니스 프로세스 언어들의 최 상위 개념만을 추상화시켰기 때문에, 좀 더 일관되고 적응적인 특징을 갖고 비즈니스 프로세스를 디자인할 수 있다.

6. 참고 문헌

[1] WfMC, Workflow Process Definition Language - XML Process Definition Language Version 1.0 Final Draft, October 25, 2002.

[2] BPMI.org, Business Process Modeling Notation (BPMN) Version 1.0, May 3, 2004.

[3] JGraph : http://www.jgraph.com/

[4] Jena : www.hpl.hp.com/semweb/jena2.htm

[5] Robert Shapiro. A Technical Comparison of XPDL, BPML and BPEL4WS, 2002.