http://dx.doi.org/10.7236/JIIBC.2013.13.1.63
JIIBC 2013-1-9

모델변환을 이용한 비즈니스 프로세스 프레임워크 5레이어 모델 자동 구축 방안

Automatic 5 Layer Model construction of Business Process Framework(BPF) with M2T Transformation

서채연*, 김영철**

Chae-Yun Seo, R. Youngchul Kim

요 약 기존 연구에서는 비즈니스 프로세스 프레임워크에서 정보 추출 및 검색을 위해 비즈니스 프로세스 시스템 질의 언어(BPSQL)를 제안했고, 기존 질의(Query)언어를 그대로 사용하기 위해 비즈니스 프로세스 프레임워크 내 각 레이어 정보들을 테이블화했지만, 레이어 정보의 스펙을 가지고 데이터베이스 구축을 수작업하는 단점이 있다. 이런 문제를 해결하기 위해 메타모델 기반의 모델-텍스트 변환기법을 적용하여, 5 레이어의 비즈니스 프로세스 모델 스키마 기반의 구축 자동 방법을 제안한다. 이를 위한 단계절차는 전체 구조와 데이터베이스 스키마의 메타 모델 정의, 모델 변형 규칙 정의 순이다. 통합정보시스템 설계의 메타 모델링을 통한 각 레이어 스펙정의와 전체 레이어 모델정보 테이블 스키마 스펙을 정의하고, 이 두 스펙 정의를 가지고 모델-텍스트 변환기법을 통해 자동으로 전체 시스템이 구축 된다. 이를 통해서 통합정보시스템 구축이 효율적으로 될 수 있다.

Abstract In previous research, we suggested a business process structured query language(BPSQL) for information extraction and retrieval in the business process framework, and used an existing query language with the tablization for each layer within the framework, but still had a problem to manually build with the specification of each layer information of BFP. To solve this problem, we suggest automatically to build the schema based business process model with model-to-text conversion technique. This procedure consists of 1) defining each meta-model of the entire structure and of database schema, and 2) also defining model transformation rules for it. With this procedure, we can automatically transform from defining through meta-modeling of an integrated information system designed to the schema based model information table specification defined of the entire layer each layer specification with model-to-text conversion techniques. It is possible to develop the efficiently integrated information system.

Key Word : MDA, UML, Metamodel, Model Transformation, Business Process Framework(BPF), Query Language, BPSQL, ModelToText Transformation Language

Ⅰ. 서 론

빠른 시간 내 변경된 비즈니스 프로세스를 효율적이고 빠르게 개발하고자 정보공학의 프레임워크를 개선하여 비즈니스 프로세스 프레임워크를 제안하였다[7]. 프레임워크를 사용하는 이유는 다른 시간, 다른 장소, 다른 사

*회원, 홍익대학교 소프트웨어공학연구실
**정회원, 홍익대학교 컴퓨터정보통신공학과
접수일자 : 2013년 1월 17일, 수정완료 : 2013년 2월 5일
게재확정일자 : 2013년 2월 8일

Received: 17 January 2013 / Revised: 5 February 2013 / Accepted: 8 February 2013
*Corresponding Author: bob@selab.hongik.ac.kr
Dept. of Computer Information & Comm., Hongik University

람이 만든 시스템을 쉽게 통합할 수 있기 때문이다.

급변하는 비즈니스 프로세스를 효율적으로 변경하기 위한 비즈니스 프로세스 프레임워크를 개발하였다[1].

비즈니스 프로세스 프레임워크 내 정보를 저장하고, 변경이 필요 할 때, 필요한 정보들만 변경하여 새로운 프로세스가 만들어진다. 각 정보를 자동으로 저장하고, 필요한 정보를 얻기위해 SQL문이 필요하다. 우리는 제안한 BPF에서 필요한 정보를 얻기위한 쿼리언어인 BPSQL를 제안했다[10]. BPF에서 각 레이어 정보는 데이타베이스 테이블 형태로 저장된다. 테이블화를 위하여 레이어 스펙은 테이블 속성이 된다. 각 레이어 스펙은 데이터베이스 테이블의 속성으로 변경하는 수작업이 필요하다. 그러나 각 레이어 정보는 하나의 정보가 아닌 많은 레이어정보가 존재한다. 그래서, 본 논문에서는 각 레이어 정보를 자동으로 테이블에 저장하기위한 방법을 연구한다. 비즈니스 프로세스 프레임워크 레이어의 정보 검색 및 접근하기 위한 BPSQL(Business Process Structured Query Language)를 개발하였다[8]. 비즈니스 프로세스 프레임워크 내 SQL을 사용하기위해 레파지토리는 테이블화하여 정보를 저장한다. 레파지토리에 정보를 저장하기위해서는 BPF 5-레이어의 복잡한 구조를 분석해야한다. 그러나, BPF의 메타모델을 통해 각 레이어를 모델링하고, 정보를 XMI (XML Metadata Interchange)로 변환하면 정보를 쉽고 효과적으로 레파지토리에 저장이 가능하다.

본 논문의 구성은 다음과 같다. 2장은 관련연구, 3장은 비즈니스 프로세스 프레임워크와 비즈니스 프로세스 프레임워크 메타모델, 4장은 메타모델을 이용한 BPF 설계와 XMI, Medel to Text Transformation Language를 통한 SQL 자동생성, 각 레이어 모델정보 자동 테이블 생성, 결론 및 향후 연구를 기술한다.

II. 관련 연구

1. Metamodel

객체 지향 모델 작성에 사용되는 UML 메타 모델의 필수 요소와 문법, 구조를 정의하는 메타 모델. 객체 및 컴포넌트 기술의 핵심을 정형화한 모델로서 MDA에서 메타 모델 정의 언어(MOF)는 CWM(Common Warehouse Metamodel)이나 UML(Unified Modeling Language)의 메타 모델에 대한 공통 모델로서 제공된다. MOF의 정형화된 내용은 MDA 모델들을 위한 표준 저장소의 역할을 수행한다[9].

2. XMI(XML Metadata Interchange)

확장성 생성 언어(XML) 기반 데이터 관리를 위한 표준으로써 메타 모델 정의 언어(MOF) 기반 모델을 XML로 매핑하기 위한 표준 명세. 구축된 모델이나 메타 모델들을 유용하게 사용하기 위해서는 툴과 툴 사이 혹은 툴과 저장 장소 사이에 이동 가능해야 하는데, 이것을 객체관리 그룹(OMG)에서 표준화한 것이다[13]. XML 메타 데이터 교환(XMI)은 문서형 정의(DTD)와 스키마를 통해 XML로 메타 모델과 모델의 표현을 표준화한다. XMI DTD와 스키마는 모델들에 대한 메타 데이터가 된다. XMI는 XML의 태그가 MOF 기반의 모델을 XML로 표현하는 데 어떻게 사용되는지를 정의한다. MOF 기반의 메타 모델은 XML DTD로 번역되며, 모델은 XML 문서로 번역된다. XMI는 객체와 객체 간 관계를 나타내기 위해 태그 기반의 언어를 사용한다[9].

3. M2T(Model To Text Transformation Language)

텍스트 변환 언어로 MOF 모델 (Mof2Text 또는 MOFM2T)는이 객체 관리 그룹 에 대한 (OMG)이 사양 모델 변환 언어 . 특히,이 예를 들어, 텍스트 (M2T)에 모델을 변환 변환 표현하는 데 사용할 수있는 플랫폼 특정 모델을 에 소스 코드 또는 문서를 . MOFM2T는 OMG의 의 한 부분입니다 모델 중심 아키텍처 (MDA)와 많은 개념 재사용 MOF , OMG의 metamodeling 아키텍처를. MOFM2T는 M2T 변환을 표현하는 데 사용됩니다 반면, OMG의 QVT(Qquery View Transformation)는 M2M 변환을 표현하는 데 사용됩니다.

4. Nested SQL(Nested structured query language)

구조적 질의어 SQL은 데이터베이스에서 정보를 얻거나 갱신하기 위한 표준화된 언어로서 대화형으로 이용하거나, 또는 프로그램 내에 삽입하여 쓸 수 있다.

중첩 쿼리는 주 쿼리의 결과를 필터링한다. 중첩 된 쿼리는 일부 데이터베이스 관리자가 "하위 쿼리"라고 한다. 메인 쿼리 결과가 반환 된 후 중첩 된 쿼리가 실행된다. 중첩 된 쿼리는 일반적으로 매우 큰 데이터를 필터링한

다. 이러한 유형의 쿼리는 현재 테이블 내에서 레코드를 필터링에 사용된다. 중첩 쿼리는 외부 테이블에서 레코드 목록을 검색하여 결과를 필터링한다.

Ⅲ. 비즈니스 프로세스 프레임워크

1. 비즈니스 프로세스 프레임워크 5-레이어

그림 1은 비즈니스 프로세스 프레임워크는 5-레이어 구조이다.

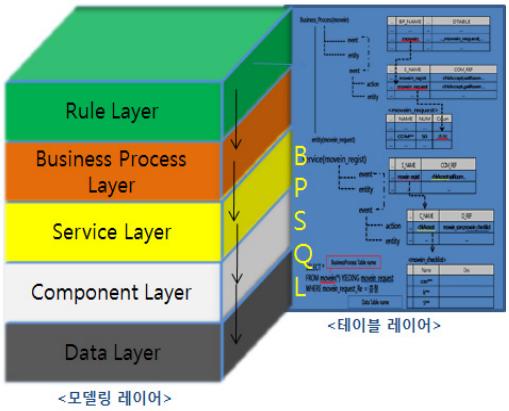

그림 1. 비즈니스 프로세스 프레임워크 5-레이어
Fig. 1. Business Process Framework 5-Layer

1레이어- 비즈니스 규칙
2레이어- 비즈니스 프로세스
3레이어- 서비스
4레이어- 컴포넌트
5레이어- 데이터베이스이다.

각 레이어별 레파지토리가 있다[5]. BPF는 인접한 상, 하위 계층이 직접 연결된 클로즈 구조이다[7].

레이어 구조는 비즈니스 요구 발생 시 이미 존재하는 재사용 컴포넌트를 활용하여 신속하게 신규 서비스를 생성할 수 있으며, 이러한 서비스로 신규 비즈니스 프로세스를 생성할 수 있다[5]. 각 레이어의 레파지토리는 테이블화하여, BPSQL를 통해 각 계층별 데이터 쿼리를 생성하고 필요한 정보를 추출한다[6].

2. BPSQL(Business Process System Query Language)

비즈니스 프로세스 프레임워크 내 레이어 모델링 정보는 테이블화하여 레파지토리에 저장한다. 저장된 정보는 재사용가능하다. 레이어 모델링 정보는 각 상세스펙에 맞게 데이터베이스에 저장된다. 테이블화되어 저장된 모델링 정보는 비즈니스 프로세스 시스템 쿼리 언어(BPSQL)을 사용하여 필요한 정보를 얻을 수 있다.

표는 비즈니스 프로세스 프레임워크에서 사용 할 수 있는 BPSQL의 문법이다. 사용 할 수 있는 쿼리언어는 테이블 생성 구문 "CREATE", 데이터베이스 내 정보를 검색하는 구문 "SELECT" 이다. 비즈니스 프로세스 프레임워크에서 검색어는 Nested 형태를 사용한다.

<nested query expression> :=
{ IN <process name><query expression>}

표 1. 비즈니스프로세스프레임워크 질의 언어 문법
Table 1. BPF Query Langauge Notation

```
<query expression> := <retrieval specification clause>
        <object reference clause>
        [ <search condition clause> ]
        [ <nested query expression> ]

<retrieval specification clause> := 'SELECT' <retrieval spec>

<retrieval spec> := '*'
        | 'BOOL'
        | <functions>
        | <attributes>

<object reference clause> := 'FROM' <data object references>
        | 'FROM' <component object references>
        | 'FROM' <service object references>
        | 'FROM' <process object references>
        | 'FROM' <rule object references>
        | 'FROM' <process id> ( <data id> )
        | 'FROM' <component id>( <data id> )
        | 'FROM' <service id> ( <component id> )
        | 'FROM' <process id> ( <service id> )
        | 'FROM' <rule id> ( <process id> )
        |'FROM' <data object reference> IN ROLE OF<data
object reference>

<data object reference> := [ <instance variable> ':' ] <data id>
<component object references> := [ <data object reference> ':'
] <component id>
<service object references> := [ <component object reference>
':' ] <service id>
<process object references> := [ <service object reference> ':'
] <process id>
<rule object references> := [ <rule object reference> ':' ] <rule
id>

<nested query expression> := { IN <process name><query
```

```
expression>}
<process name>:= policy | business process|service|component|data

<search condition clause> := 'WHERE' <search-constraint>

<search-constraint> := <boolean expression>
        | <non_temporal constraint>

<non_temporal constraint> := <functions>
        | <condition predicates>

<condition predicate> := <expression> <op> <expression> [ <uncertainty operator> ]

<op> := <comp op> | <logical op> | <quantifier>
<comp op>:== '<>' | '<' | '<=' | '>='
<logical op> := 'AND' | 'OR' | 'NOT'
<quantifier> := 'SOME' | 'NO' | 'ALL' | 'EVERY' | 'ONLY' | 'SAME' | 'THIS'
        | 'ATLEAST' <number> | 'AT MOST' <number> | 'EXACTLY' <number>
<uncertainty operator> := 'ACCEPTABLE' | 'PREFERRED' | 'HIGH' | 'LOW' | 'MEDIUM'

<functions> := <aggregate functions>
        | <user-defined functions/methods>
<aggregate functions> := 'MAX' | 'MIN' | 'AVG' | 'SUM' | 'COUNT'
<user-defined functions/methods> := <function/method identifier>
```

위의 표현은 Nested 형태의 검색어를 사용하기 위한 문법이다.

Ⅳ. 메타모델을 이용한 BPF레이어 설계와 XMI

기본적인 BPF구조를 메타모델화하고 메타모델에 맞개발하는 시스템을 모델링한다. 비즈니스 프로세스 프레임워크의 메타모델은 개발하는 시스템을 5레이어로 모델링하고, 모델링한 정보를 자동으로 시스템에 저장하기 위함이다. 모델은 시스템에 저장할 수 없기 때문에, 모델정보를 XMI로 바꾸고, XMI를 모델-변환 언어를 이용, 각 레이어 모델정보를 데이터베이스에 자동으로 저장 할 수 있다. 즉, 그림 3과 같이 설계를 통해 데이터베이스에 자동 구축하기 위한 방법을 언급한다.

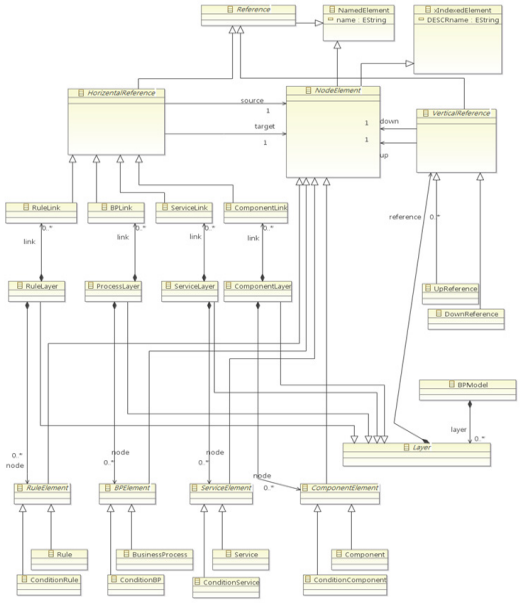

그림 2. 비즈니스 프로세스 프레임워크 메타 모델
Fig. 2. Business Process Framework Metamodel

1. BPF Metamodel

비즈니스 프로세스 프레임워크의 각 레이어를 모델링 가능한 노테이션이 존재한다.

그 노테이션을 비즈니스 프로세스 프레임워크 모델링 노테이션 언어(BPFMN : Business Process Framework Modeling Notation)라 한다[1]. BPFMN으로 각 레이어를 모델링한다. 그림 통합정보관리시스템 사업을 모델링한 것이다. 비즈니스 프로세스 프레임워크 메타 모델을 이용하여 BPFMN로 메타모델한다. 그림 2는 비즈니스 프로세스 프레임워크의 메타 모델이다. 비즈니스 프로세스 프레임워크 메타 모델을 기반으로 통합정보관리시스템의 사업프로세스를 메타 모델한다.

2. 메타모델을 이용한 레이어 모델링

비즈니스 프로세스 프레임워크의 메타모델을 하기위해 이클립스의 EMF(Eclips Modeling Framework)를 사용했다. EMF 도구를 사용하여 5-레이어 메타 모델을 설계한다. 그림 3은 사업프로세스를 관리하기 위한 통합정보관리시스템이다. 메타모델 형식으로 통합정보관리시스템을 설계한다. 설계한 정보는 XMI로 변환 할 수 있다.

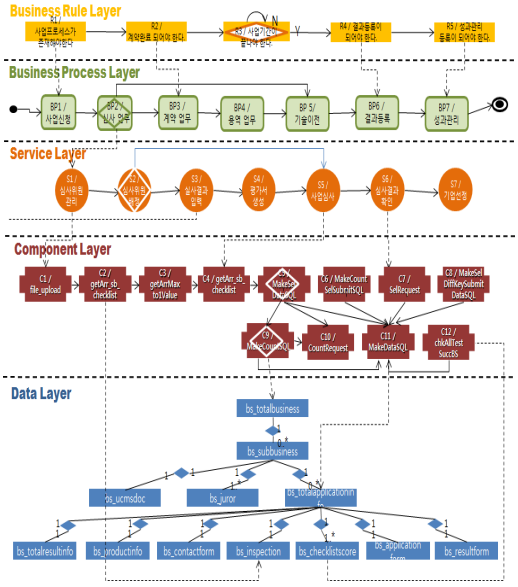

그림 3. 통합정보관리시스템 사업 프로세스 BPFMN 모델
Fig. 3. Integrated information management system model, a business process BPFMN

3. 통합정보관리시스템 XMI

표 2는 통합정보관리시스템의 사업을 메타 모델링한 XMI다. 룰, 비즈니스 프로세스, 서비스, 컴포넌트는 "node", 조건이 있는 룰, 비즈니스 프로세스, 서비스, 컴포넌트는 "condition ~"으로 표현한다. 동일레이어에 연결된 선은 "link"로 표시하고, "source"와"target"으로 입력과 출력이 구분된다. 상/하위 레이어에 연결된 선은 UpReference/DownReference로 표시된다.

표 2. 통합정보관리시스템 사업프로세스의 XMI
Table 2. IntegratedInformationManagementSystem XMI

```
<?xml version="1.0" encoding="UTF-8"?>
<bpmetamodel:BPModel xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:bpmetamodel="http://bpmetamodel/1.0">
  <layer xsi:type="bpmetamodel:RuleLayer">
    <reference xsi:type="bpmetamodel:DownReference" name="R1"
down="//@layer.1/@node.0"/>
    <reference xsi:type="bpmetamodel:DownReference" name="R2"
down="//@layer.1/@node.1"/>
    <reference xsi:type="bpmetamodel:DownReference" name="R4"
down="//@layer.1/@node.5"/>
    <reference xsi:type="bpmetamodel:DownReference" name="R5"
down="//@layer.1/@node.6"/>
    <node xsi:type="bpmetamodel:Rule" name="R1" DESCRname="Being
BusinessProcess"/>
    <node xsi:type="bpmetamodel:Rule" name="R2"/>
    <node xsi:type="bpmetamodel:ConditionRule" name="R3"/>
    <node xsi:type="bpmetamodel:Rule" name="R4"/>
    <node xsi:type="bpmetamodel:Rule" name="R5"/>
    <link name="r1" source="//@layer.0/@node.0"
target="//@layer.0/@node.1"/>
    <link name="r2" source="//@layer.0/@node.1"
target="//@layer.0/@node.2"/>
    <link name="r3" source="//@layer.0/@node.2"
target="//@layer.0/@node.3"/>
    <link name="r4" source="//@layer.0/@node.2"
target="//@layer.0/@node.2"/>
    <link name="r5" source="//@layer.0/@node.3"
target="//@layer.0/@node.4"/>
  </layer>
  <layer xsi:type="bpmetamodel:ProcessLayer">
    <reference xsi:type="bpmetamodel:DownReference" name="BP2"
up="//@layer.0/@node.0" down="//@layer.2/@node.0"/>
    <node xsi:type="bpmetamodel:BusinessProcess" name="BP1"/>
    <node xsi:type="bpmetamodel:BusinessProcess" name="BP3"/>
    <node xsi:type="bpmetamodel:ConditionBP" name="BP2"/>
    <node xsi:type="bpmetamodel:BusinessProcess" name="BP4"/>
    <node xsi:type="bpmetamodel:BusinessProcess" name="BP5"/>
    <node xsi:type="bpmetamodel:BusinessProcess" name="BP6"/>
    <node xsi:type="bpmetamodel:BusinessProcess" name="BP7"/>
    <link name="bp1" source="//@layer.1/@node.0"
target="//@layer.1/@node.2"/>
    <link name="bp2" source="//@layer.1/@node.2"
target="//@layer.1/@node.1"/>
    <link name="bp2'" source="//@layer.1/@node.2"
target="//@layer.1/@node.4"/>
    <link name="bp3" source="//@layer.1/@node.1"
target="//@layer.1/@node.3"/>
    <link name="bp4" source="//@layer.1/@node.3"
target="//@layer.1/@node.4"/>
    <link name="bp5" source="//@layer.1/@node.4"
target="//@layer.1/@node.5"/>
    <link name="bp6" source="//@layer.1/@node.5"
target="//@layer.1/@node.6"/>
  </layer>
  <layer xsi:type="bpmetamodel:ServiceLayer">
    <reference xsi:type="bpmetamodel:DownReference" name="S1"
up="//@layer.1/@node.2" down="//@layer.3/@node.0"/>
    <reference xsi:type="bpmetamodel:DownReference" name="S2"
down="//@layer.3/@node.9"/>
    <reference xsi:type="bpmetamodel:DownReference" name="S3"
down="//@layer.3/@node.10"/>
    <reference xsi:type="bpmetamodel:DownReference" name="S4"
down="//@layer.3/@node.1"/>
    <reference xsi:type="bpmetamodel:DownReference" name="S5"
down="//@layer.3/@node.2"/>
    <reference xsi:type="bpmetamodel:DownReference" name="S6"
down="//@layer.3/@node.4"/>
    <reference xsi:type="bpmetamodel:DownReference" name="S7"
down="//@layer.3/@node.8"/>
    <node xsi:type="bpmetamodel:Service" name="S1"/>
    <node xsi:type="bpmetamodel:ConditionService" name="S2"/>
    <node xsi:type="bpmetamodel:Service" name="S3"/>
    <node xsi:type="bpmetamodel:Service" name="S4"/>
    <node xsi:type="bpmetamodel:Service" name="S5"/>
    <node xsi:type="bpmetamodel:Service" name="S6"/>
    <node xsi:type="bpmetamodel:Service" name="S7"/>
    <link name="s1" source="//@layer.2/@node.0"
target="//@layer.2/@node.1"/>
    <link name="s2" source="//@layer.2/@node.1"
target="//@layer.2/@node.2"/>
    <link name="s2'" source="//@layer.2/@node.1"
target="//@layer.2/@node.4"/>
    <link name="s3" source="//@layer.2/@node.2"
target="//@layer.2/@node.3"/>
    <link name="s4" source="//@layer.2/@node.3"
target="//@layer.2/@node.4"/>
    <link name="s5" source="//@layer.2/@node.4"
target="//@layer.2/@node.5"/>
    <link name="s6" source="//@layer.2/@node.5"
target="//@layer.2/@node.6"/>
  </layer>
  <layer xsi:type="bpmetamodel:ComponentLayer">
    <reference xsi:type="bpmetamodel:DownReference" name="C2"/>
    <reference xsi:type="bpmetamodel:DownReference" name="C9"/>
    <reference xsi:type="bpmetamodel:DownReference" name="C12"/>
    <reference xsi:type="bpmetamodel:DownReference" name="C8"/>
    <reference xsi:type="bpmetamodel:UpReference" name="C1"
up="//@layer.2/@node.0"/>
    <reference xsi:type="bpmetamodel:UpReference" name="C3"
up="//@layer.2/@node.4"/>
    <reference xsi:type="bpmetamodel:UpReference" name="C4"
up="//@layer.2/@node.4"/>
```

```
    <reference xsi:type="bpmetamodel:UpReference" name="C7"
up="//@layer.2/@node.5"/>
    <reference xsi:type="bpmetamodel:UpReference" name="C9"
up="//@layer.2/@node.6"/>
    <reference xsi:type="bpmetamodel:UpReference" name="C11"
up="//@layer.2/@node.2"/>
    <node xsi:type="bpmetamodel:Component" name="C1"
DESCRname=""/>
    <node xsi:type="bpmetamodel:Component" name="C2"/>
    <node xsi:type="bpmetamodel:Component" name="C3"/>
    <node xsi:type="bpmetamodel:Component" name="C4"/>
    <node xsi:type="bpmetamodel:ConditionComponent" name="C5"/>
    <node xsi:type="bpmetamodel:Component" name="C6"/>
    <node xsi:type="bpmetamodel:Component" name="C7"/>
    <node xsi:type="bpmetamodel:Component" name="C8"/>
    <node xsi:type="bpmetamodel:ConditionComponent" name="C9"/>
    <node xsi:type="bpmetamodel:Component" name="C10"/>
    <node xsi:type="bpmetamodel:Component" name="C11"/>
    <node xsi:type="bpmetamodel:Component" name="C12"/>
    <link name="c1" source="//@layer.3/@node.0"
target="//@layer.3/@node.1"/>
    <link name="c2" source="//@layer.3/@node.1"
target="//@layer.3/@node.2"/>
    <link name="c3" source="//@layer.3/@node.2"
target="//@layer.3/@node.4"/>
    <link name="c4" source="//@layer.3/@node.3"
target="//@layer.3/@node.4"/>
    <link name="c5" source="//@layer.3/@node.4"
target="//@layer.3/@node.10"/>
    <link name="c5'" source="//@layer.3/@node.4"
target="//@layer.3/@node.9"/>
    <link name="c5''" source="//@layer.3/@node.4"
target="//@layer.3/@node.8"/>
    <link name="c6" source="//@layer.3/@node.5"
target="//@layer.3/@node.10"/>
    <link name="c7" source="//@layer.3/@node.6"
target="//@layer.3/@node.10"/>
    <link name="c8" source="//@layer.3/@node.7"
target="//@layer.3/@node.10"/>
    <link name="c9" source="//@layer.3/@node.8"
target="//@layer.3/@node.9"/>
    <link name="c9'" source="//@layer.3/@node.8"
target="//@layer.3/@node.10"/>
    <link name="c11" source="//@layer.3/@node.10"
target="//@layer.3/@node.11"/>
    <link name="c12" source="//@layer.3/@node.11"
target="//@layer.3/@node.10"/>
  </layer>
</bpmetamodel:BPModel>
```

표 2 XMI데이타를 자동분석하여 모델-텍스트 변환언어에 룰을 적용하여 각 레이어 모델 정보는 테이블화한다. 그림 4는 “CREATE” 구문으로 각 레이어 모델링 스키마 정보를 생성한다.

그림 5는 RuleTable 스키마 정보로 통합정보시스템 사업 룰을 데이터베이스 구축한다.

	name	DESCRname	link	DownReference
1	R1	customer Inquires	R2	P1
2	R2	registration	R3	P2
3	R3	customer Account Representative	R2,R4	P3
4	R4	Oder Dispatcher	R5	P4
5	R5	Delivery Dispatcher	R6	P5
6	R6	Delivery Item	R4,R7	P6
7	R7	Close Order	R8	P7

그림 5. RuleTable 데이터
Fig. 5. RuleTable Data

그림 6은 ProcessTable 스키마 정보에 의해 생성된 데이터베이스다.

	name	DESCRname	link	DownReference	UpReference
1	P1	customer Inquires	P2		R1
2	P2	registration	P3	S1,S2,S3,S4,S5	
3	P3	customer Account Representative	P2,P4	S6,S7,S8,S9	R2
4	P4	Oder Dispatcher	P5	S10,S11,S12,S13	
5	P5	Delivery Dispatcher	P6	S14,S15,S16,S17	R4
6	P6	Delivery Item	P4,P7	S18,S19,S20,S21	R5
7	P7	Close Order			R7

그림 6. ProcessTable 데이터
Fig. 6. ProcessTable Data

그림 7은 ServiceTable 스키마에 의해 생성된 통합정보시스템 사업의 서비스 테이블이다.

	name	DESCRname	link	DownReference	UpReference
1	S1	Customer Inquires	S2		P1
2	S2	Customer Application	S3		
3	S3	Good Credit	S4		R2
4	S4	Order	S5	S10,S11,S12,S13	
5	S5	Customer Account ...	S6	S14,S15,S16,S17	R4
6	S6	Order Item	S7	S18,S19,S20,S21	R5
7	S7	View Order	S8		R7

그림 7. ServiceTable 데이터
Fig. 7. ServiceTable Data

그림 8은 ComponentTable 스키마에 의해 생성된 통합정보시스템의 컴포넌트 테이블이다.

	name	DESCRname	link	DownReference	UpReference
1	C1	customer Inquires	C2		R1
2	C10	Accepted Order	C8		R7
3	C11	Warehouse Personal	C9		R7
4	C12	Item Found	C10		R7
5	C2	registration	C3	S1,S2,S3,S4,S5	
6	C3	customer Account Representative	C2,C4	S6,S7,S8,S9	R2
7	C4	Oder Dispatcher	C5	S10,S11,S12,S13	
8	C5	Delivery Dispatcher	P6	S14,S15,S16,S17	R4
9	C6	Delivery Item	C4,C7	S18,S19,S20,S21	R5
10	C7	Close Order	C5		R7
11	C8	Order Acceptable	C6		R7
12	C9	Order Dispatcher	C7		R7

그림 8. ComponentTable 데이터
Fig. 8. ComponentTable Data

"INSERT ~ INTO"는 각 테이블에 데이터를 저장한다. “SELECT”는 검색한 정보를 얻을 때 사용한다.

이때, Nested 형태의 SQL구문을 사용하여 각 레이어 정보를 얻는다. 비즈니스 프로세스 프레임워크를 메타모델하고, 메타모델을 기반으로 각 레이어를 모델링한다. 모델링한 레이어 정보는 모델변환하여 XMI로 저장한다. XMI 데이터는 ModelToText 변환언어를 사용하여 자동으로 SQL구문을 발생하고 레파지토리를 테이블 형태로 자동 저장한다. 본 논문에서는 지면상으로 그림 3메타 모델링 정보에서 그림 5~그림 8로 데이터베이스 자동 구축되는 방법만 언급하고, 모델 변형 규칙에 대한 직접적인 설명은 제외한다.

V. 결 론

급격히 변화하는 비즈니스 환경에서 한 기업의 비즈니스 프로세스 프레임워크을 효율적으로 구축하려는 방안으로, 메타모델 기반의 모델-텍스트 변환기법[12]을 적용하여, 5 레이어의 비즈니스 프로세스 모델 스키마 기반의 구축 자동 방법을 언급하였다. 이는 전체 구조와 데이터베이스 스키마의 메타 모델 정의, 모델 변형 규칙 정의 등의 절차이며 통합정보시스템 설계의 메타 모델링을 통한 각 레이어 스펙정의와 전체 레이어 모델정보 테이블 스키마 스펙을 정의하고, 이 두 스펙 정의를 가지고 모델-텍스트 변환기법을 통해 자동으로 전체 시스템이 구축된다.

향후 연구는 메타모델 기반의 모델-텍스트 변환 언어를 사용하여 XMI정보와 테이블 스키마를 매핑 할 수 있는 룰을 만들어 자동 테이블 발생 연구를 진행 할 것이다.

참 고 문 헌

[1] Chae Yun Seo, Dong Woo Kim, R. Young Chul Kim, “Business Process Framework based on the Closed Architecture”, Journal of the Korea Academia-Industrial cooperation Society, Vol. 10, No. 8, 1939-1946, Aug 2009.

[2] Chae Yun Seo, Dong Woo Kim, R. Young Chul Kim, “5-Layer Architecture for Efficient Business Process Modeling”,IWIT, Vol. 6, No. 1 19-22 May 2008.

[3] James Martin, “Information Engineering”, Prentice-Hall International, Inc, 1990

[4] Chae Yun Seo, So Young Moon, R. Young Chul Kim, Byoung-Ho Ahn, “A Study on Modeling Efficient Business Process Framework: Mapping Business process Layer and Data Layer”,The 1st YEllow Sea International Conference on ubiquitous Computing, Shandong Univ, China Vol. 1, 59 Aug 2011.

[5] Roger S. Pressman ″Software Engineering A Practitioners’ Approach″ 3rd Ed, McGraw Hill, 2004.

[6] par Hedley Apperly, Ralph Hoffman, et Steve Latchem “Service-And Component-Based Development″

[7] Chae Yun Seo, So Young Moon, R. Young Chul Kim, “A Study on Data Migration for BPSQL Query Language on Business Process Framwork (BPF)”, KIISE, Sep 2011.

[8] Woo Yeol Kim, Hyun Seung Son, R. Young Chul Kim, “Development of Windows Mobie Applications using Model Transformation Techiques” KIISE, Jun 2010.

[9] Chae-Yun Seo, R. Young Chul Kim, “Development through Mapping CBD, Service Model onto BPM based on Business Process Framework”, KIPS, Apr 2012.

[10] Chae-Yun Seo, R. Young Chul Kim, “Design of Metamodel for 5 Layer Information on Business Process Framework”, KIPS, Nov 2012.

[11] K. Czarnecki, S. Helsen, ″Feature-based survey of model transformation approaches″ Mar 2006.

[12] OMG, ″MOF Model to Text Transformation Language, v1.0 Feb 2008.

[13] Dong-Kuk Ryu, R. Young Chul Kim, ″Design and Implementation of M&S Based Test Environment for Interoperability Verification of Heterogeneous Composite Embedded System, Journal of Korean Institute of Information Technology, vol. 7, issue 2, pp. 33-40, Apr 2009.

※ 본 연구는 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업(2012-0001845)과 교육과학기술부와 한국연구재단의 지역혁신인력양성사업으로 수행된 연구결과임.

저자 소개

서 채 연(회원)

- 2005년 2월 : 홍익대학교 일반대학원 전자전산과 (공학석사)
- 2008년 2월 : 홍익대학교 일반대학원 전자전산과 (박사수료)
- 2009년 3월 ~ 현재 : 유한대학 경영정보 겸임교수

<주관심분야 : 메타모델, 비즈니스 프로세스 모델, 컴포넌트, 서비스, 프레임워크>

김 영 철(회원)

- 2000년 : Illinois Institute of Technology (공학박사)
- 2000년 ~ 2001 : LG산전중앙연구소 Embedded system 부장
- 2001년 ~ 현재 : 홍익대학교 컴퓨터정보통신 교수

<주관심분야 : 테스트 성숙도 모델 (TMM), 임베디드 소프트웨어 개발 방법론, 모델 기반 테스팅, 메타모델, 비즈니스 프로세스 모델, 사용자 행위 분석 방법론>