Journal of the Korea Academia-Industrial cooperation Society
Vol. 16, No. 10 pp. 6969-6976, 2015

http://dx.doi.org/10.5762/KAIS.2015.16.10.6969
ISSN 1975-4701 / eISSN 2288-4688

메타모델 기반의 방법론 프레임워크 설계

조은숙[1*]
[1]서일대학교 컴퓨터소프트웨어과

Design of Methodology Framework based on Meta-Model

Eun-Sook Cho[1*]
[1]**Dept. of Computer Software, Seoil University**

요 약 새로운 기술의 발전과 개발 패러다임의 변화로 인해 이를 뒷받침할 수 있는 소프트웨어 개발 프로세스와 개발 기법들이 계속해서 진화하고 있다. 이러다보니 기업에서 프로젝트 별로 개발 및 적용해야 할 프로세스와 방법론이 계속해서 증가하고 있어서 이에 대한 효과적인 관리 방안이 필요한 실정이다. 특히 기업은 프로젝트의 규모나 성격에 따라 최적화 된 방법론을 도입해서 적용해야 하기 때문에 방법론의 특화(Customization) 기법이 절실히 요구되는 상황이다. 본 논문에서는 이처럼 계속해서 개발되는 방법론들을 전사적으로 통합 관리할 수 있는 메타모델 기반의 방법론 프레임워크를 제시하고자 한다. 제안된 방법론 프레임워크를 적용함으로써 기업에서는 계속해서 증가되는 방법론들을 효율적으로 관리할 수 있을 뿐만 아니라 프로젝트 사안 별로 최적화 된 방법론을 용이하게 개발할 수 있게 된다. 특히 본 논문에서 제시하는 방법론 프레임워크는 메타 모델을 기반으로 개발되기 때문에 새로운 방법론 요소의 추가 및 확장이 용이하게 이루어질 뿐만 아니라 방법론을 프로젝트 별로 특화할 때 쉽게 구성요소들을 재사용할 수 있는 효과를 얻게 된다.

Abstract As new technologies are advancing and development paradigms are changing, software development process and development methods are evolving progressively. As a result, because the number of developing and managing processes and methodologies are increasing as a project in companies, effective management methods are needed. Especially, because companies should apply optimized methodology according to project's size and characteristics, customization technique of methodology is required urgently. In this paper, we propose a meta-model based methodology framework which can integrate and manage methodologies being developed progressively. Applying proposed methodology framework, a company is able to manage as well as develop optimized methodology easily as a project. Especially, because a proposed methodology framework is developed by meta-model, adding or extending new methodology elements can be realized simply as well as method elements are reused easily in case of customization of methodology as a project.

Keywords : Development Process, Methodology Framework, Meta-Model, Customization

1. 서론

소프트웨어 기술이 발전함에 따라 소프트웨어의 규모나 복잡도 또한 점차 증가하고 있는 실정이다. 이뿐만 아니라 소프트웨어의 이용 범위가 정보처리와 같은 수준에서 사물 인터넷(IoT: Internet of Things) 수준으로 확대되고 있는 상황이다[1]. 이렇게 많은 변화들이 등장함에 따라 소프트웨어를 개발하는 개발 패러다임의 변화가 일어나고 이를 뒷받침할 수 있는 소프트웨어 개발 방법론과 개발 프로세스가 계속해서 진화 발전하고 있는 추세이다. 소프트웨어 개발 방법론은 1970년대 소프트웨어 위기가 대두되면서 등장하기 시작하였고, 이후로 계속

본 논문은 2015년 서일대학교 교내연구과제로 수행되었음.
*Corresponding Author : Eun-Sook Cho(Seoil University)
Tel: +82-10-8870-4595 email: escho@seoil.ac.kr
Received August 3, 2015 Revised August 31, 2015
Accepted October 8, 2015 Published October 31, 2015

기술의 발전과 더불어 끊임없이 새로운 방법론들이 연구되고 발전되어 오고 있다. 1970년대에는 메인 프레임 컴퓨팅 환경에서 구조적 개발 방법론이 등장하였고, 1980년대는 클라이언트/서버 환경으로 변하면서 정보공학 방법론이 등장하였고, 1990년대는 인터넷과 웹 기반의 환경으로 인해 객체지향 방법론이 등장하였다[2,3][18]. 2000년대 소프트웨어 재사용 기술에 대한 관심이 증가하면서 컴포넌트 기반 개발 방법론이 등장하였고, 2010년대 유비쿼터스 컴퓨팅 환경과 모바일 컴퓨팅 환경 그리고 사물 인터넷의 등장으로 서비스 기반 개발 방법론, 모바일 개발 방법론, 임베디드 개발 방법론 등이 등장하였다[4,5][15]. 지금도 계속해서 새로운 컴퓨팅 환경의 변화에 따른 개발 방법론들이 연구되고 개발되고 있는 실정이다. 이처럼 기술의 발전으로 방법론들이 계속해서 등장함으로서 기업에서는 개발된 방법론들을 전사적으로 통합 관리할 수 있는 체계를 갖고 있지 않다. 이로 인해 기업에서 프로젝트 별로 프로젝트의 규모나 성격에 최적화 된 방법론과 개발 프로세스를 구축하기가 매우 어렵다[17]. 이는 결국 프로젝트 수행에 있어서 많은 불편함과 예기치 못한 오류를 발생 시키게 된다. 그리고 이미 개발된 방법론의 구성 요소들을 효율적으로 재사용하지 못함으로써 방법론의 효용성 가치를 저하시키는 결과를 초래하게 된다[6]. 본 논문에서 이러한 문제점을 해결하기 위해 기업 내의 방법론들을 전사적으로 통합 관리할 수 있는 개발 방법론 프레임워크를 제시하고자 한다. 프로세스 프레임워크를 통해 산재되어 있는 방법론들을 하나로 통합 관리하고 또한 프로젝트 별로 필요한 방법론 구성요소들을 조립하여 최적화 된 방법론을 쉽게 개발 및 적용할 수 있도록 하고자 한다. 특히 본 논문에서는 재사용성과 확장성을 고려하여 메타모델 기반의 개발 방법론 프레임워크를 제시하고자 한다. 메타모델 기법을 적용함으로써 구성 요소들의 중복성은 회피하면서 확장성을 용이하게 할 수 있다.

본 논문의 구성으로 2장에서는 관련연구로 메타 모델 언어인 MOF(Meta Object Facility)[7]와 SPEM(Software & system Process Engineering Meta-Model) 2.0[8]에 대해 설명하며, 3장에서는 본 논문에서 제안하는 개발 방법론 프레임워크의 아키텍처와 개발 방법론 프레임워크의 메타 모델과 그 구성 요소들을 정의하여 제시한다. 4장에서는 3장에서 제안한 메타모델을 적용한 실제 모델 사례에 적용한 결과를 제시한다. 마지막으로 5장에서 결론 및 향후 연구 과제를 제시한다.

2. 관련 연구

소프트웨어는 계획, 분석, 설계, 구현, 테스팅, 유지보수와 같은 과정들을 거쳐서 개발되게 된다. 이러한 개발 과정을 소프트웨어 개발 프로세스라 하며, 여기에 조직 구조와 개발 기법이 반영된 것을 개발 방법론이라 한다. 본 논문에서는 이러한 개발 프로세스에 대한 프레임워크를 메타모델을 기반으로 정의하기에 메타모델과 OMG(Object Management Group)에서 정의된 SPEM2.0에 대해 설명하고자 한다.

메타모델이란 어떤 모델에 대한 의미를 추상화하여 표현한 모델로서, 예를 들면 객체지향 모델링 언어인 UML(Unified Modeling Language)의 요소들에 대한 의미를 모델로 정의한 MOF가 메타 모델에 해당한다[8,9,10,11][16]. 본 논문에서는 이 MOF를 기반으로 프로세스 프레임워크를 정의하고자 한다.

MOF는 객체 및 컴포넌트 기술의 핵심을 정형화한 모델로서 MDA에서 MOF는 CWM(Common Warehouse Meta-Model)이나 UML의 메타모델에 대한 공통 모델로서 제공된다. MOF의 정형화된 내용은 MDA 모델들을 위한 표준 저장소의 역할을 수행한다.

OMG(Object Management Group)의 프로세스 메타 모델인 SPEM 2.0은 Fig.1과 같은 구조를 지니고 있다.

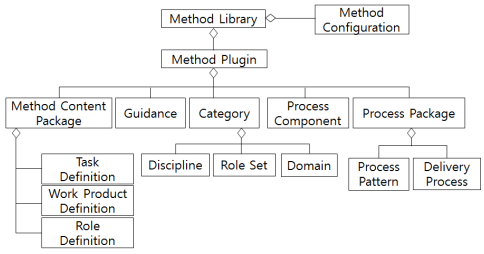

Fig. 1. SPEM 2.0 Structure

모든 SPEM 2.0의 구성요소들은 Method Library에 저장된다. Method Plugin는 Content 와 Process 패키지를 포함하는 패키지이다. Method Plugin은 메소드 컨텐츠와 프로세스의 모듈화와 조직화를 위한 입자(Granularity) 수준을 정의한다. Method Configuration은 각 메소드 플러그인의 메소드 컨텐츠 패키지와 프로세스

패키지의 부분집합이며, 선택된 Method Plugin 들의 집합체이다. Method Content Package는 Work Product Definition, Task Definition, Role Definition 등을 포함하는 패키지이다. Guidance는 Describable Element와 관련된 추가적인 정보를 제공하는 Describable Element 이다. Guidance는 guidance의 특정 타입을 나타내는 Kind로 분류된다. Guidance를 위한 Kind의 예로는 Guidelines, Templates, Checklists, Tool Mentors, Estimates, Supporting Materials, Reports, Concepts 등이 있다. Category는 사용자의 기준에 따라 컨텐츠를 분류하는데 사용될 수 있다. Process Component는 캡슐화의 원칙을 적용한 특별한 Process Package 이다. Process Component는 Activity로 표현되는 하나의 프로세스를 포함하고, Process Component를 위한 입력과 출력을 정의하는 Work Product Port의 set을 정의한다. Process Package는 Process 단위로 패키지할 수 있는 요소들을 포함할 수 있는 특별한 패키지이다. Task Definition은 Role Definition 인스턴스에 의해 수행되는 작업을 정의하는 Method Content Element이고 Work Definition 이다. Task는 입력 및 출력 Work Product와 연관된다. Work Product Definition은 Task Definition에 의해서 사용되고, 변경되고, 생성되는 Method Content Element 이다. Role은 Task를 수행하기 위해 Work Product를 사용하고, Task를 수행하는 과정에서 Work Product를 생성한다. Discipline은 관심사의 유사성과 work effort의 협력에 기반 한 작업의 분류이다. Role Set은 공통성을 가진 Role들을 그룹화한 것이다. Domain은 리소스, 타이밍, 관계를 기반으로 서로 관련된 work product 들을 논리적으로 그룹화 한 것이다. 도메인은 하위 도메인으로 나누어질 수 있다.

Tool Category는 Tool Mentor의 집합체로 도구와 도구의 기능에 대한 설명을 제공한다. Process Pattern은 공통된 문제에 대한 일관된 개발 방법을 제공하는 일반적인 프로세스 영역에 있는 재사용 가능한 Activity의 집합을 칭하는 특별한 Process 이다. Process Pattern은 Delivery Process 내에서 어디에서나 적용할 수 있도록 설계되어야 한다. Delivery Process는 특정 유형의 프로젝트를 수행하기 위해 완전하고 통합된 방법을 기술하는 특별한 Process 이다. Delivery Process는 완전한 프로젝트 생명주기(Life Cycle)을 기술하고 프로젝트를 수행하기 위해 레퍼런스로 사용된다.

SPEM 2.0기반 프로세스 저작도구인 Ossellus사의 IRIS Author[12]나 Eclipse 기반의 EPF(Engineering Process Framework)[13]에서도 행위, 산출물, 역할 등을 독립적으로 정의하고 자유롭게 조합할 수 있는 기능 들을 제공하고 있다.

3. 메타모델 기반의 방법론 프레임워크

이 장에서는 메타모델과 컴포넌트 기술을 기반으로 한 소프트웨어 개발 방법론 프레임워크의 아키텍처와 내부 구성 요소들에 대해 설명하고자 한다. 또한 이러한 구성 요소들을 기반으로 방법론을 어떻게 최적화시켜서 개발 프로세스에 적용시키는지 설명하고자 한다.

3.1 메타모델 기반의 개발 방법론 프레임워크

본 논문에서 정의하는 개발 방법론 프레임워크(Methodology Framework)는 소프트웨어 개발 방법론들을 정형적인 메타 모델 기반으로 저작하고 수정하고 관리하기 위한 도구이다. 제안하는 개발 방법론 프레임워크는 OMG의 SPEM 2.0을 기반으로 정의한다. 즉 SEP 메타모델을 기반으로 정의 되며, SEP 메타모델은 SPEM 2.0의 서브 클래스로서 정의 된다. 결과적으로 Fig.2에서와 같이 SEP 메타모델은 MOF, UML(Unified Modeling Language) Meta-Model 등으로 정의되는 OMG의 MDA(Model-Driven Architecture)를 기반으로 정의 된다. SPEM 2.0을 기반으로 정의됨으로서, MDA가 제공하는 모든 특징 들을 그대로 수용한다. 더욱이 UML과의 호환성도 지니게 되어 객체지향 모델링 언어가 가지는 객체지향 표현 능력을 가지게 되며, MDA 모델의 완전성, 무결성 등을 보장 받게 된다.

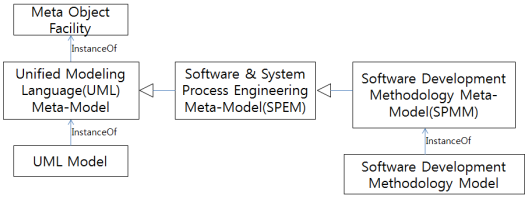

Fig. 2. Hierarchy of Development Methodology Framework

Fig. 2와 같이 개발 방법론의 메타모델은 UML 메타모델을 기반으로 정의되고, 이렇게 정의된 메타 모델을

기반으로 각각의 소프트웨어 개발 방법론이 생성된다. 이를 통해 방법론 내의 각각의 작업(Activity) 들과 산출물(Work-Product) 들 간의 일관성(Consistency), 추적성(Traceability), 재사용성(Reusability)을 향상 시킨다.

3.2 개발 방법론 프레임워크 아키텍처

개발 방법론 프레임워크(Development Methodology Framework) 아키텍처는, Fig.3과 같이 메소드 (Method), 프로세스(Process), 패러다임(Paradigm) 등 3개의 독립적인 축(Axis) 혹은 관점(View)으로 나뉘어 구성 된다.

프로세스 측면은 단계(Phase), 반복(Iteration) 등 마일스톤(Milestone)을 중심으로 순차적(Sequential) 인과관계를 가지는 구성요소 즉 프로세스 컴포넌트(Process Component)들로 구성된다. 예를 들어 Unified Process의 경우 도입(Inception), 이행(Elaboration), 구축(Construction), 전이(Transition) 등과 같은 단계들이 이에 해당한다. 마일스톤을 갖는 시간적 단위라는 측면에서 프로세스는 프로젝트 종료라는 특수한 마일스톤을 갖는다.

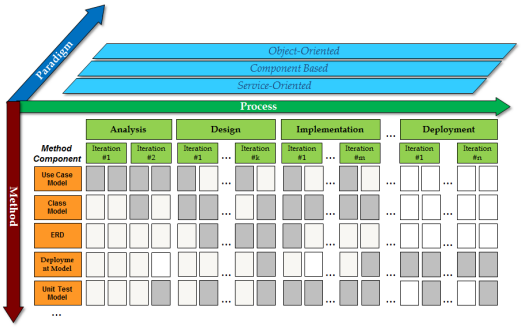

Fig. 3. Architecture of Development Methodology

메소드 측면은 방법론의 작업 산출물(Work Product)과 이를 작성하기 위한 행위(Activity)의 집합인 메소드 클래스(Method Class), 메소드 클래스들의 집합체인 메소드 컴포넌트(Method Component)로 구성된다. 메소드 컴포넌트는 프로세스 컴포넌트를 구성하는 구성 단위가 되며, 메소드 컴포넌트는 다시 메소드 클래스들로 구성된다. 예를 들어, 요구사항 모델이나 설계 모델 등과 같은 것이 메소드 컴포넌트에 해당되며, 유스케이스 다이어그램과 클래스 다이어그램 등이 이 모델을 구성하는 요소가 된다. 패러다임은 메소드를 전개하는데 있어서 접근 방법을 의미한다.

3.3 개발 방법론 메타 모델

Fig.4는 개발 방법론 메타모델을 나타내고 있다. 개발 방법론 메타모델은 앞 절의 메타모델 핵심클래스에 설명한 바와 같이 Method Content와 Process로 구성된다.

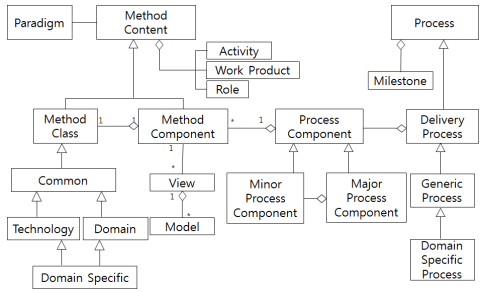

Fig. 4. Meta-model of Methodology Framework

Method Content는 작업 산출물인 Work-Product와 Activity의 집합체인 Method Class와 Method Class 들의 집합체인 Method Component로 구성된다.

Method Class는 작업 산출물을 중심으로 재사용 가능한 단위 클래스로 정의한다. Method Class는 공통, 기술공통, 업무공통 등 3가지 유형으로 분류된다. Method Component는 Method Class들의 구조체이며, 일반적으로 단위 모델로서 정의 된다. 모델은 일반적으로 관점별로 생성 되며, 독립적 작업 단위라 할 수 있다. Method Class 내의 상속 구조를 UML 모델링 도구인 Enterprise Architect(EA)[14]로 모델링한 형태는 Fig.5와 같다.

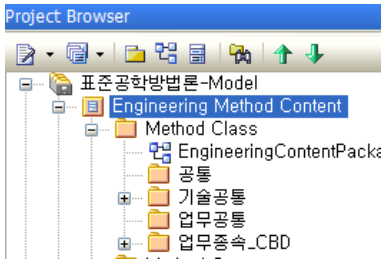

Fig. 5. Hierarchy of Method Class

Fig.6에 나타난 것처럼 Method Component는 이해당사자의 관점별로 나누어진다. 크게 비즈니스(Business) 모델 관점, 시스템(System) 모델 관점, 사용(In-Use) 모델 관점으로 분류되며, 시스템 모델은 다시 소프트웨어 모델, 하드웨어 모델로 다시 분류된다. 소프트웨어 모델은 사용자 관점의 모델과 개발자 관점의 모

델로 구분하고, 개발자 관점의 모델은 다시 플랫폼에 독립적인 모델과 플랫폼에 종속적인 모델로 분류한다. 플랫폼 종속 모델은 구현 독립 모델과 구현 종속 모델로 분류한다.

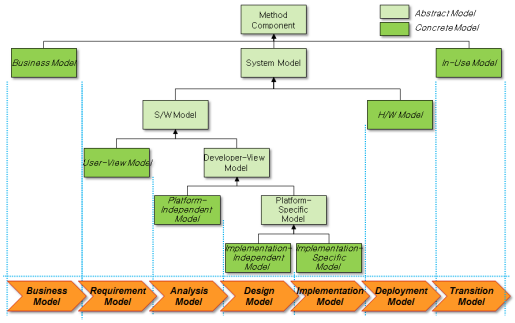

Fig. 6. Method Component

이러한 분류 원칙을 통해 개발된 메소드 컴포넌트들은 비즈니스 모델, 요구모델, 분석모델, 설계모델, 구현모델, 전개모델, 이행 모델로 구분하여 정의한다. 이러한 메소드 컴포넌트들을 EA로 모델링 한 형태가 Fig.7과 같다.

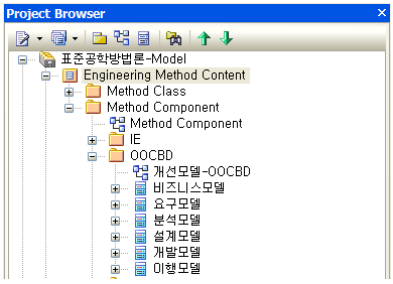

Fig. 7. Hierarchy of Method Component

Process는 마일스톤을 달성하기 위한 작업적 순서를 가진 관리적 단위이다. 프로세스 컴포넌트는 시간적 개념을 반영한 실제 수행 프로세스를 구성하는 단위 요소로서, 이미 정의된 메소드 컴포넌트들을 조립하여 구성한 복합 컴포넌트이다. 실제 여러 프로젝트 별 수행 프로세스를 정의하는 데 있어서 공통적으로 재사용할 수 있는 프로세스 단위이다. 아키텍처 중심 방법론의 예를 들면, 프로세스 컴포넌트는 크게 비즈니스 모델링, 아키텍팅(Architecting), 구축, 이행 단계로 구성하여 정의한다. 각각의 프로세스 컴포넌트는 하나 이상의 메소드 컴포넌트들로 구성된다. Process의 유형으로 Process Component와 Delivery Process가 있다. Process Component는 Method Component들을 작업 순서별로 구성해 구축한다. Process Component의 마일스톤은 Major Process Component와 Minor Process Component로 구분 된다. 위의 프로세스 컴포넌트를 EPF로 모델링 한 결과가 Fig.8과 같다. Fig.8에 표현된 것처럼 EPF에서는 프로세스 컴포넌트를 Capability Pattern으로 정의를 한다. 그리고 Capability Pattern 안에는 여러 다양한 프로세스 컴포넌트들을 정의해서 실제 프로젝트 별 수행 프로세스에서 재사용할 수 있도록 정의한다.

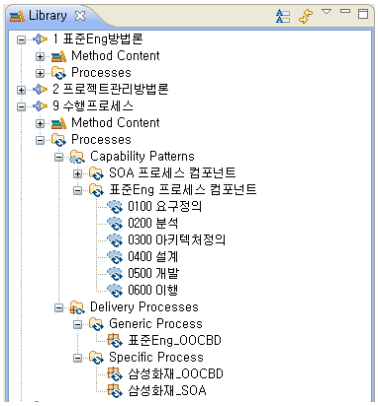

Fig. 8. Hierarchy of Process Component

Delivery Process는 Process Component를 작업 순서별로 조합하여 생성한다. Delivery Process는 재사용을 위해 Generic Process, Domain Specific Process 로 계층화 한다. Fig.9는 특정 수행 프로세스의 프로세스 컴포넌트 및 메소드 컴포넌트 구조를 나타내고 있으며, Fig.10은 이를 EPF에 구현한 사례를 보여주고 있다.

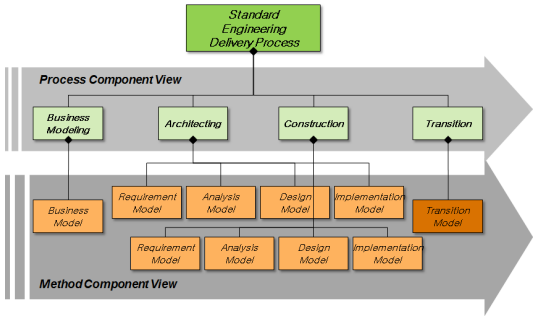

Fig. 9. Architecture of Process Component & Method Component in Delivery Process

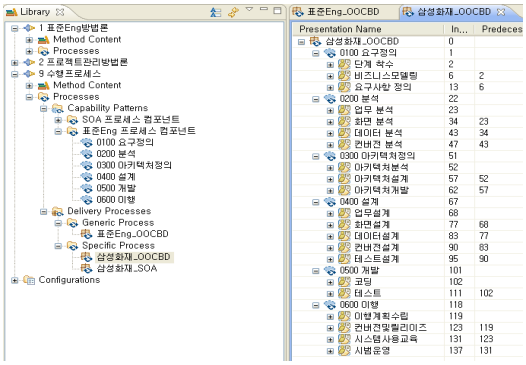

Fig. 10. Hierarchy of Delivery Process

4. 사례연구

이 장에서는 앞 장에서 제시한 메타모델 프로세스 프레임워크를 기반으로 실제 S사에 적용한 사례 연구를 제시 하고자 한다.

4.1 메소드 클래스 정의

S사의 방법론에서는 이전 장에서 설명한 개념을 토대로 메소드 클래스를 크게 4가지 패키지인 공통, 기술공통, 업무 공통, 업무 종속으로 분류하였다.

첫째, 공통 패키지는 특정 기술이나 업무에 독립적으로 사용되는 산출물들로 구성되어 있다.

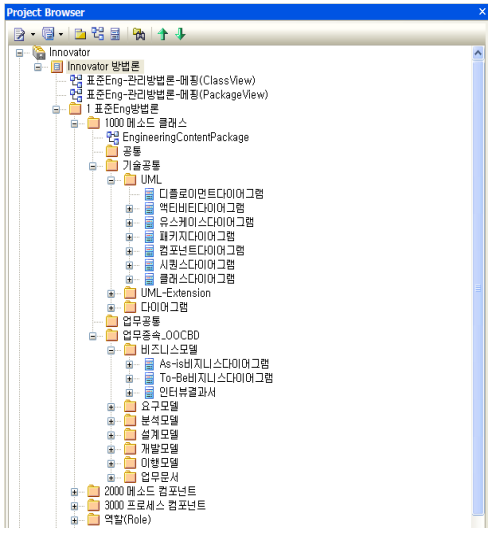

Fig. 11. S Company's Method Class

둘째, 기술 공통 패키지는 특정 기술과 관련된 산출물들로 구성되어 있다. Fig.11에 제시된 것처럼 S사의 방법론에서는 UML에서 제공하는 다이어그램들과 UML을 확장한 산출물, 그리고 기타 ERD와 같은 산출물들을 사용하고 있기 때문에 기술 공통 패키지 안에는 다시 서브 패키지로 UML 패키지와 UML-Extension 패키지, 그리고 다이어그램 패키지로 구분하여 정의하고 있다.

셋째, 업무 공통 패키지는 특정 업무에 공통적으로 재사용 가능한 산출물들로 구성되어 있다.

넷째, 업무 종속 패키지는 특정 업무와 패러다임(예: CBD, IE, SOA 등등)에 종속적인 산출물들로 구성된다. S사의 방법론에서 사용하는 업무 종속 산출물들을 다시 업무 특성 별로 분류하였는데, Fig.12에 제시된 것처럼 비즈니스 모델, 요구모델, 분석 모델, 설계모델, 개발모델, 이행모델, 그리고 업무 문서 등으로 정의된다. 예를 들어 분석 모델의 경우에는 분석 업무와 관련된 산출물들이 클래스로 정의되어 있다.

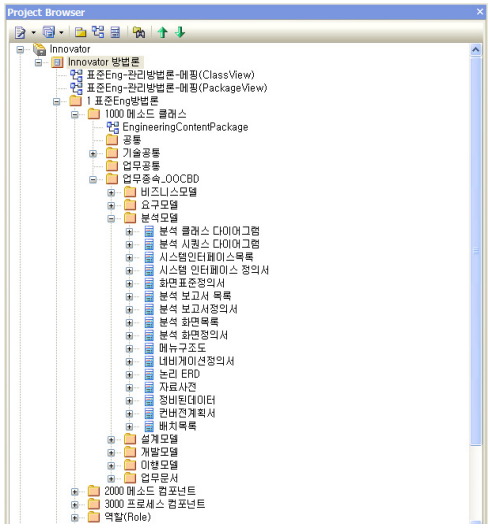

Fig. 12. S Company's Domain Specific Package

4.2 메소드 컴포넌트

메소드 컴포넌트는 패러다임에 따라 IE 메소드 컴포넌트 패키지와 CBD 메소드 컴포넌트 패키지 또는 SOA 메소드 컴포넌트 패키지 등이 될 수 있으며, 각 패키지에는 Fig. 13과 같이 비즈니스 모델, 요구모델, 분석모델, 설계모델, 개발모델, 이행모델 등의 복합 클래스 형태로 표현한다. 이 클래스는 위에 정의된 메소드 클래스의 클

래스가 될 수도 있고 여러 메소드 클래스들을 복합적으로 갖는 또 다른 복합 클래스가 될 수도 있다.

예를 들어, Fig. 13에 나타난 것처럼 OOCBD 메소드 컴포넌트 내에는 여러 개의 복합 클래스인 비즈니스 모델, 요구 모델, 분석 모델, 설계 모델, 개발 모델, 이행 모델 등이 있다.

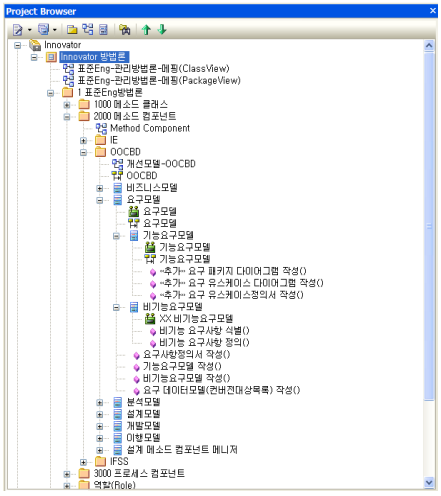

Fig. 13. S Company's Method Component

4.3 프로세스 컴포넌트

S사 방법론의 프로세스 컴포넌트는 크게 요구 정의, 분석, 아키텍처 정의, 설계, 개발, 이행 컴포넌트로 분류하여 정의하였다. Fig. 14는 프로세스 컴포넌트의 종류(요구정의, 분석, 아키텍처 정의 등등)들이 열거 되어 있으며, Fig. 15는 각 프로세스 컴포넌트 별 활동과 내부 메소드 컴포넌트들이 표현되어 있다.

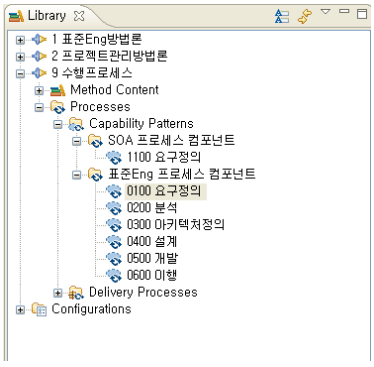

Fig. 14. S Company's Process Component

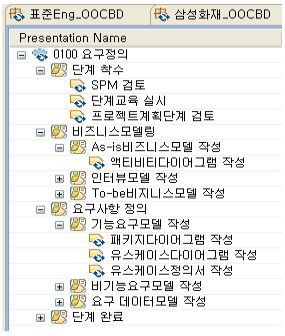

Fig. 15. S Company's Activity and Method Component per Process Component

4.4 수행 프로세스

수행 프로세스는 실제 프로젝트에 반영하는 프로세스를 의미한다. 수행 프로세스는 다시 두 가지 카테고리로 분류할 수 있다. 한 가지는 도메인에 공통된 수행 프로세스인 Generic 프로세스이고 다른 한 가지는 실제 프로젝트 단위 별 실행되는 Specific 프로세스이다.

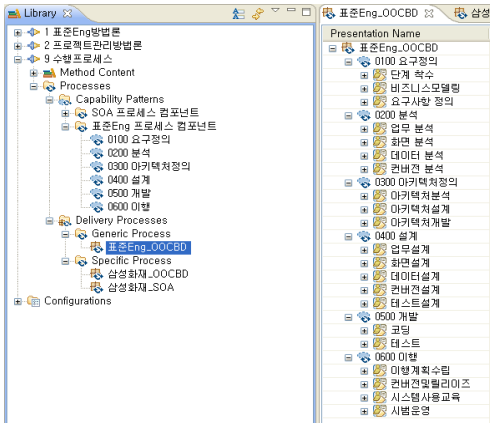

Fig. 16. S Company's Delivery Process

여기서는 이미 정의된 프로세스 컴포넌트들을 기반으로 재사용하여 수행 프로세스를 정의한다. 여기서는 이미 정의된 프로세스 컴포넌트들을 갖다가 조합하여 추가적으로 필요한 부분들을 추가하거나 확장해서 정의한다. 이 경우는 이미 정의된 것들을 대부분 재사용하기 때문에 프로세스 정의가 매우 용이하다고 할 수 있다. Fig.16에 제시된 것처럼 S사의 표준 Eng_OOCBD 프로세스는 프로세스 컴포넌트인 표준 Eng 프로세스 컴포넌트를 기반으로 복사하여 작성하였기 때문에 표준 Eng_OOCBD의 프로세스 컴포넌트가 그대로 재사용되었다.

5. 결론 및 향후 연구과제

기술의 발전과 컴퓨팅 환경의 변화에 발맞춰 소프트웨어 개발 프로세스와 방법론 또한 계속 발전하고 새롭게 진화하고 있다. 앞으로도 계속하여 IT기술이 발전함에 따라 이에 따른 소프트웨어 개발 프로세스 및 방법론 또한 발전할 것이다. 이렇게 계속해서 증가하는 프로세스와 방법론을 보다 체계적이면서 효율적으로 관리하기 위한 방안으로 본 논문에서 메타 모델 기반의 개발 방법론 프레임워크를 제시하였다.

제시된 프레임워크의 효율성을 검증해 보기 위해 국내 굴지의 기업인 S사에 본 프레임워크를 적용해 보았고 그 결과 기존에 해당 기업에 산재해 있는 방법론들을 체계적으로 통합할 수 있었고 이를 통해 프로젝트 사안 별로 여러 방법론의 구성 요소들을 조립하여 용이하게 프로젝트에 최적화된 개발 방법론과 프로세스를 생성할 수 있었다. 또한 이를 지원하는 EA와 EPF와 같은 도루를 활용함으로써 손쉽게 조립이 가능할 수 있었다. 향후 연구 과제로는 제시한 메타모델의 요소들을 검증하고, 새롭게 신생하는 방법론의 요소들을 확장하여 발전시키고 이를 지원할 수 있는 자동화 도구를 구축하는 것이다.

References

[1] Dave Evans, "The Internet of Things: How the Next Evolution of the Internet Is Changing Everything", *CISCO White Paepr*, April 2011.

[2] Bansler, J., Bodker, K.. *A reappraisal of structured analysis: Design in an organizational context*. ACM Trans. Information Systems, pp.165-193, 1993. DOI: http://dx.doi.org/10.1145/130226.148055

[3] Desmond Francis Dsouza, Alan Cameran wills, *Objects, Component, and Frameworks with UML: the Catalysis approach* , Addison Wesley, 1999.

[4] Alfonso Fuggetta, "Software process: a roadmap." *Proceedings of the Conference on The Future of Software Engineering*, p.25 34, June 04 11, 2000, Limerick, Ireland, 2000. DOI: http://dx.doi.org/10.1145/336512.336521

[5] Salmre, I., *Writing Mobile Code: Essential Software Engineering for Building Mobile Applications*, Addison-Wesley Professional, 2005.

[6] Winston W. Royce, "Managing the Development of Large Software Systems." *In Proceedings of IEEE WESCON*, p 1 - 9, August 1970.

[7] Object Management Group. MDA Documents. http://www.omg.org/docs/omg/03-06-01.pdf, 2003.

[8] OMG, Software & Systems Process Engineering Metamodel specification (SPEM) Version 2.0, http://www.omg.org/spec/SPEM/2.0/PDF, 2008.

[9] Erich Gamma et. al, *Design Patterns-Elements of Reusable Object-Oriented Software*, Addison-Wesley, 2012.

[10] Craig Larman, *Applying UML and Patterns*, Prentice Hall, 2001.

[11] Martin Fowler, *UML Distilled*, Addison-Wesley,2003.

[12] IRIS Author, http://files.osellus.com, 2010.

[13] EPF, http://www.eclipse.org/epf, 2010.

[14] EA, http://www.sparxsystems.com/products/ea. 2015.

[15] Sattarova Feruza, Farkhod Alisherov, "Effective Estimation Software cost using Test Generations," *Asia-pacific Journal of Multimedia Services Convergent with Art, Humanities, and Sociology,* Vol. 1 No.1, pp. 1-10, Jun. 2011. DOI: http://dx.doi.org/10.14257/AJMAHS.2011.06.01.

[16] Jung Hwan Par, Do Hyeun Kim, Su Kyung Yang, Eun Hyeon Koh, "A Practicum Training Model for Early Childhood Pre-Service Teacher Using UML," Asia-pacific Journal of Multimedia Services Convergent with Art, Humanities, and Sociology, Vol. 4 No. 2, pp.2 23-234, Dec. 2014. DOI: http://dx.doi.org/10.14257/AJMAHS.2014.12.22

[17] Byeong Ho Knag, Tai-hoon Kim, "Effective Optimization of Multi-Clouds using Software-as-a-Service," *Asia-pacific Journal of Multimedia Services Convergent with Art, Humanities, and Sociology*, Vol. 2 No. 2, pp.8 5-92, Dec. 2012. DOI: http://dx.doi.org/10.14257/AJMAHS.2012.12.04

[18] Godfrey D. Beray, Sabah Mohammed, "Software engineering and their conflicts in development of software," *Asia-pacific Journal of Multimedia Services Convergent with Art, Humanities, and Sociology*, Vol. 3 No. 1, pp.9-16, Jun. 2013. DOI: http://dx.doi.org/10.14257/AJMAHS.2013.06.05

조 은 숙(Eun-Sook Cho) **[정회원]**

- 1993년 2월 : 동의대학교 전산통계학과 졸업(이학사)
- 1996년 2월 : 숭실대학교 대학원 컴퓨터학과(공학석사)
- 2000년 2월 : 숭실대학교 대학원 컴퓨터학과(공학박사)
- 2000년 3월 ~ 2005년 2월 : 동덕여자대학교 정보학부 강의전임교수
- 2005년 3월 ~ 현재 : 서일대학교 컴퓨터 소프트웨어과 부교수

<관심분야>

Software Engineering, Software Development Methodology, Mobile Computing, IoT