객체지향 환경에서의 ISO/IEC 12207 소프트웨어 생명주기 공정 표준에 대한 적용성 연구

김은영°, 백인섭
아주대학교 컴퓨터공학과

A Study on Adaptability of ISO/IEC 12207 Software Life Cycle Process Standard in Obejct-Oriented Environment

Eun-Young Kim°, In-Sup Paik
Dept. of Computer Engineering, Ajou University

요 약

소프트웨어 시스템에 대한 요구사항이 증가하고 그 규모가 점차 거대·복잡해짐에 따라 시스템을 개발하기 위한 절차, 방법 등이 무수히 생겨나게 되었다. ISO/IEC 12207 소프트웨어 생명주기 공정 표준은, 존재하고 있는 소프트웨어 개발 및 유지보수에 대한 절차 및 방법에 대한 일반적인 지침을 제시하고, 이전의 프로세스 공정 표준들이 적용될 수 없는 새로운 개발 패러다임을 지원하기 위해 제정되었다. 이 표준은 표준 자체의 특성의 하나인 보편성을 유지하기 위해, 특정 프로세스 모델에 치우치지 않는 기반 표준(Base Standard)의 성격을 갖는다. 결국 기반표준은, 어떠한 프로젝트에서 어떠한 프로세스 모델을 사용하건, 임의의 방법론을 사용하든 모든 경우에 적용될 수 있어야 한다. 최근, 질적인 성장 뿐 아니라 양적으로도 풍부한 성장을 거둔 객체지향 개발 환경에서는 전통적인 개발 환경에서와 많은 차이점을 보이며, 새로운 개발 프로세스들을 제안하고 있다. 본 논문에서는 객체지향 개발 프로세스 중 현재 가장 대중적으로 사용되고 있는 Rational사의 Unified Process를 선택하여 ISO/IEC 12207표준의 적용성을 고찰해보고, 표준의 발전, 개선방향에 대해 모색해보겠다.

1. 서론

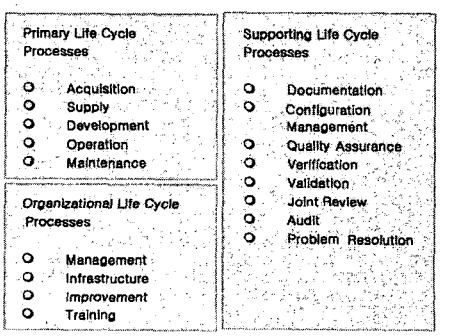
소프트웨어 시스템의 규모가 커지고 복잡도가 증가함에 따라 많은 문제점이 나타나게 되었으며, 이를 해결하기 위하여 다수의 절차, 방법, 도구 등이 출현하였다. 그러나 이것은 한편으로 소프트웨어 시스템 개발에 혼란을 가중시키게 되었다. 결국 통일된 지침을 제시하는 소프트웨어 공정 표준의 필요성이 대두되었고, 1980년대 이후 여러 표준들이 통합과 발전을 거듭하여 1995년 ISO/IEC 12207 소프트웨어 생명주기 공정 표준(Software Life Cycle Process Standard)이 국제 표준으로 제정된 후 현재까지 사용되고 있다[1][2].

또한 ISO/IEC 12207 표준이 새롭게 제정된 주요 이유중의 하나는 기존에 존재했던 표준들이 새로운 개발 패러다임인 객체지향 방식이나 RAD(Rapid Application Development) 방식에 적용하는 과정에서 이전에서는 나타나지 않은 새로운 문제점들이 발생했기 때문이다[3].

객체지향에 대한 기본 아이디어는 1980년대 중반 이후에 이미 도입이 되었지만, 초기에는 객체지향 언어 및 프로그래밍에 대한 연구가 주 대상이었다. 1990년대에 들어서면서 성숙된 객체지향 언어를 바탕으로 한 객체지향 기반 프로세스 모델들이 개발되기 시작하였고(OOSE, Objectory Process), 이후 Rational 사에서는 이러한 프로세스 모델들을 통합, 보완하는 과정을 거쳐 *Rational Unified Process* (이하 RUP)로 발전시켰으며, 현재 객체지향을 대표하는 프로세스 모델로 인식되고 있다. 실제 ISO/IEC 12207 표준이 객체지향의 개념을 수용하고 있다고는 하나, 이러한 사실을 검증해 볼 수 있는 프로세스 모델이 그 당시에는 미약한 수준이었고, 기반 표준(Base Standard)으로써 사용되기 위해서는 현재 대중적으로 사용되고 있는 프로세스 모델에 대해서도 그 적용성을 검증받아야 한다. 따라서 본 논문에서는 현재 소프트웨어 생명주기 공정 분야의 국제 표준인 ISO/IEC 12207 소프트웨어 생명주기 공정 표준을 객체지향 프로세스 모델인 *Rational Unified Process*에서 적용, 검증해보고, 이를 바탕으로 표준의 개선, 발전방향에 대해 모색해보고자 한다.

2. ISO/IEC 12207 소프트웨어 생명주기 공정 표준 [4]

ISO/IEC 12207 소프트웨어 생명주기 공정 표준은 1995년에 소프트웨어 개발을 위한 일괄적이고 체계적인 구조(framework)를 제공하기



[그림 1] ISO/IEC 12207 표준의 구조

위해 제정되었다. 이 표준은 소프트웨어 개발 시 고려해야 할 5개의 기본(Primary) 프로세스, 8개의 지원(Supporting) 프로세스, 4개의 조직(Organizational) 프로세스에 대한 내용으로 구성된다. 표준을 이루고 있는 각 프로세스는 다음과 같다 [그림 1]. 각 프로세스들은 하나 이상의 액티비티(Activity)들로 이루어지며, 액티비티는 하나 이상의 태스크(Task)들로 이루어진다.

> 기본 생명주기-프로세스 (Primary Life Cycle Process)

소프트웨어 개발 프로세스의 주 골격을 이루는 프로세스들이다. 소프트웨어의 획득, 공급, 개발, 운영, 유지보수에 대한 활동을 정의한다.

> 지원 생명주기-프로세스 (Supporting Life Cycle Process)

기본 프로세스들이 원활히 진행될 수 있도록 보조해주는 역할을 하는 프로세스이다. 각 기본 프로세스로부터 산출되는 문서화, 품질 보증, 감사, 문제해결 등에 대한 활동을 정의한다.

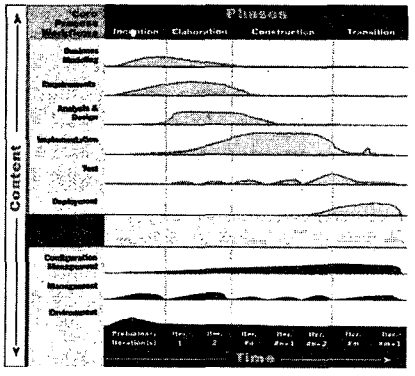
> 조직 생명주기-프로세스 (Organizational Life Cycle Process)

개발 전주기에 거쳐 전체 프로젝트를 관리하는 역할을 하는 프로세스이다. 프로젝트의 기반구조, 개선, 인력훈련 등에 대한 활동을 정의한다.

본 표준은 보편성을 유지하면서 특정 프로세스 모델에 치우치지 않는 기반표준(Base Standard)이기 때문에, 소프트웨어 개발에 필요한 요소적 프로세스들을 전체적으로 보여주고 있지만 실제 프로세스들이 어떻게 상호 연관되는지, 프로세스 내의 액티비티 및 태스크들이 어떻게 상호 연관되는 지에 대해 명시하고 있지 않다. 결국 이 표준을 이용해 소프트웨어 개발을 수행하는 조직들은 개발하려는 프로젝트의 성격에 맞도록 방법론 및 프로세스 모델을 선정 혹은 정립하고, 프로세스 모델의 세부 절차들은 본 표준에서 제시된 항목들을 취사선택, 변형(tailoring)해야 하는 기능화(functionality) 작업을 거쳐야 한다.

3. Rational Unified Process[5][6]

RUP는 가장 대중화된 객체지향 개발 프로세스 중의 하나이다. RUP는 기존의 프로세스 모델과는 달리 2차원으로 표현된다[그림 2].



[그림 2] Rational Unified Process

하나의 차원([그림 2]의 세로축)은 핵심 워크플로우(Core Workflows)로써, 프로세스 모델을 구성하는 요소적 프로세스들에 대한 정적 특성을 표현하고, 이것은 핵심 프로세스 워크플로우(Core Process Workflows : 비즈니스 모델링, 요구사항, 분석 및 설계, 구현, 테스트, 전이)와 핵심 지원 워크플로우(Core Supporting Workflows : 형상 및 변화관리, 프로젝트 관리, 환경)로 나뉜다. 다른 한 차원은 시간에 대한 축으로써, 한 프로세스에 대한 동적 특성을 표현하는데, 이것은 다음의 네 단계로 구성된다[7][8].

> 개념화 단계 (Inception Phase)

시스템에 대한 비즈니스 사례(Business Case)를 만들고, 프로젝트 범위(Scope)를 정의하는 단계이다. 액터(Actor)와 유즈케이스(Usecase)의 추출, 성공기준(Success Criteria) 위험관리(Risk Management)에 대한 계획수립 등이 이 단계에 속한다.

> 상세화 단계 (Elaboration Phase)

문제 영역을 분석하고 아키텍처를 구성하는 단계이다. 프로젝트의 위험요소를 제거하는 것도 이 단계에서 행해져야 한다.

> 구축 단계 (Construction Phase)

반복 및 점증적(iterative and incremental)인 방법을 통해 소프트웨어를 완전히 개발하는 단계이다. 구현을 끝내고 테스트하는 과정도 여기에 포함된다.

> 전이 단계 (Transition Phase)

완성된 소프트웨어를 사용자에게 전달하는 단계이다. 사용자의 요구사항을 체크하고 문제점이 있다면 수정을 한다.

각 프로세스에서 4개의 단계는 각 단계의 마지막에 위치하는 milestone을 만족하느냐에 여부에 따라 여러번 반복되어 행해질 수 있다.

4. ISO/IEC 12207 표준의 적용 및 적용성 고찰

본 절에서는 RUP의 요소들을 바탕으로 현재 국제 표준으로 사용되고 있는 ISO/IEC 12207 소프트웨어 생명주기 공정 표준의 적용성에 대해 고찰해 본다. 표준의 적용성을 프로세스, 액티비티, 태스크 수준으로 상세화 하면서 각각의 수준별로 비교 평가 하였으며, 그 각각의 결과는 다음과 같다.

4.1 최상위 수준에서의 비교 - ISO/IEC 12207의 프로세스 vs. RUP의 워크플로우

이 수준에서의 비교 대상은 ISO/IEC 12207의 생명주기 프로세스(Life Cycle Process)내의 각 프로세스들과 RUP의 워크플로우들이다. ISO/IEC 12207 표준은 총 3가지 생명주기 프로세스와 17개의 요소적 프로세스들로 구성된다[그림 1]. 두 대상의 매핑 결과, RUP의 핵심 프로세스 워크플로우를 구성하고 있는 비즈니스 모델링, 요구사항, 분석 및 설계, 구현, 테스트, 전이는 모두 기본(Primary) 생명주기 프로세스 내의 개발(Development) 프로세스에 매핑됨을 살펴볼 수 있었다. 또한 핵심 워크플로우를 구성하고 있는 형상 및 변화관리 워크플로우는 지원(Supporting) 생명주기 프로세스 내의 형상관리(Configuration Management) 프로세스에, 프로젝트 관리와 환경 워크플로우는 조직(Organizational) 생명주기 프로세스 내의 관리(Managmenet) 프로세스와 기반(Infrastructure) 프로세스에 각각 매핑 되었다.

이러한 결과로 미루어 보아, ISO/IEC 12207 표준을 최상위 수준에서 RUP에 매핑 시키는 데에는 부족한 면이 보이지 않았다. 오히려, 프로젝트의 범위(scope)면에서 살펴보았을 때, RUP에서 기본 소프트웨어 개발을 통한 프로세스 외에 자체 프로젝트를 관리하고 지원해주는 프로세스는 표준의 총 12개 프로세스 중 3개 밖에 존재하지 않았다. 따라서 최상위 수준에서 RUP에 대한 ISO/IEC 12207표준의 적용성은 문제가 없지만, RUP가 복잡한 시스템의 개발을 위해서는 좀 더 다양한 지원 프로세스를 가져야 함이 요구되는 것을 볼 수 있다.

4.2 중간 수준에서의 적용성 비교 -

ISO/IEC 12207의 액티비티 vs. RUP의 프로세스 워크플로우

이 절에서는 RUP의 핵심 프로세스 워크플로우(Core Process Workflows)와 ISO/IEC 12207의 액티비티를 비교함으로써 표준의 적용성을 고찰해본다. [그림 1]에 명시된 5개의 프로세스 워크플로우는 ISO/IEC 1220표준에서 기본(Primary) 프로세스 내의 개발(Development) 프로세스에 해당된다. 이 개발 프로세스는 총 13개의 액티비티로 구성되어 있고, RUP의 각 프로세스 워크플로우에 매핑시킨 예는 [표 1]과 같다.

RUP의 프로세스 워크플로우	ISO/IEC 12207 개발 프로세스의 액티비티
요구사항 수집 (Requirement)	5.3.1 Process Implementation
분석 및 설계 (Analysis & Design)	5.3.4 Software Requirements Analysis 5.2.5 Software Architectural Design 5.2.6 Software Detailed Design
구현 (Implementation)	5.3.7 Software Coding and Testing 5.3.8 Software Integration 5.3.10 System Integration
테스트 (Test)	5.3.9 Software Qualification Testing 5.3.11 System Qualification Testing
전이 (Deployment)	5.3.12 Software installation 5.3.13 Software acceptance support

[표 1] RUP 프로세스 워크플로우와 ISO/IEC12207 표준의 매핑 예

[표 1]의 결과 단순히 RUP의 프로세스 워크플로우들에 ISO/IEC 12207 표준의 요소들을 적용시키는 데에는 별 문제가 없어 보인다. 그러나 이것을 개념의 정확성(conceptual completion)에서 살펴보았을 때, 문제가 발생함을 알 수 있다. RUP에서의 '아키텍처'는 전체 시스템을 구성하는 요소들을 추출하고 그 요소들 간의 관계를 정의하는 최상위 수준의 구조를 말한다[9]. 하지만 ISO/IEC 12207에서의 아키텍처는 어떤 하나의 대상을 상세하게 설계하기 전에 큰 부분으로 쪼개어 대강의 윤곽만 잡는 것을 뜻한다. 결국, 아키텍처에 대한 의미상의 차이로 인하여 ISO/EC 12207 표준을 RUP에 적용시키는데 문제가 발생된다. 특히 RUP는 아키텍처 중심(Architecture Centric)의 모델이기 때문에 이러한 문제는 치명적일 수 있다. 따라서, ISO/IEC 12207 표준을 RUP에 적용시키는 데에 아키텍처의 표현 및 구현에서 문제가 발생함을 알 수 있고, 현재 표준안으로써 연구가 진행중인 아키텍처 표준[10]과 ISO/IEC 12207 표준을 병용하여 위의 문제점을 개선해 나가야 할 필요성이 요구된다.

4.3 최하위 수준에서의 적용성 비교 -

ISO/IEC 12207표준의 태스크 vs. RUP의 워크플로우 내의 각 단계(phase)

RUP가 다른 기존의 프로세스 모델과 다른 점은 2차원 구조로 이루어졌다는 점이다. 따라서 RUP의 정적인 차원 외에 동적인 차원에 대해서도 ISO/IEC 12207 표준의 적용성을 고려할 필요가 있다. 본 절에서는 ISO/IEC 12207 표준과 RUP에서 가장 작은 단위인 태스크와 프로세스 워크플로우 내의 각 단계(Phase)를 비교함으로써 그 적용성을 고찰해본다.

분석 및 설계 (Analysis & Design)	
Inception	5.3.4.1(Document software requirements) 5.3.5.1(Transform requirements into architecture)
Elaboration	5.3.4.1 5.3.5.1 5.3.5.2(Document top-level design for interface) 5.3.5.3(Document top-level design for database) 5.2.6.1(Document design for each component) 5.2.6.2(Document design for interface) 5.2.6.3(Document design for database)
Construction	5.3.4.1 5.3.5.2 5.3.6.1 5.3.6.2 5.3.6.3
Transition	

[표 2] RUP의 단계(Phase)와 ISO/IEC12207 표준의 태스크 매핑 예

[표 2]는 RUP의 분석 및 설계 프로세스 워크플로우의 각 단계 당 매핑되는 ISO/IEC 12207의 태스크를 나열한 예를 보인다. 표에서 점으로 분리된 각 번호는 태스크 번호를 의미하며, 괄호 안에는 그 태스크의 내용을 요약하여 표기하였다. 두 번 이상 사용되는 태스크는 내용 없이 번호만을 표기하였다.

같은 워크플로우 내의 서로 다른 단계들은 이전 단계의 것을 정련화(refine) 하거나, 이전에 하지 못했던 작업을 새롭게 추가(add)하는 방법으로 되풀이되기 때문에, ISO/IEC 12207 표준의 하나의 태스크는 두 개 이상의 단계(Phase)에 중복으로 사용된다. 비록 반복되는 워크플로우 당 단계들은 그 기능이 유사하긴 하지만 완전히 같은 작업을 되풀이하는 것은 아니다. 따라서 하나의 ISO/IEC 12207 표준의 태스크를 여러 단계에 중복 사용하는 것은 기반 표준으로써 이 표준을 적용하는데 그 의미상의 정확성을 잃게 된다. 이러한 사실을 바탕으로 ISO/IEC 12207 표준이 기반표준으로써 RUP와 같은 2차원 구조를 갖는 프로세스 모델에 적용이 되려면 정적인 워크플로우 차원뿐 아니라 시간에 따라 변하는 동적 변화에 대한 고려가 이루어져야 함을 알 수 있고, 이것을 위해 하나의 태스크는 시간을 고려한 하위 태스크(SubTask)들로 세분화되어야 할 것이다.

5. 결론 및 향후 연구과제

본 논문에서는 현재 소프트웨어 생명주기 공정 국제 표준으로 사용되고 있는 ISO/IEC 12207 표준이 객체지향 소프트웨어 개발 환경에서 얼마나 적용성이 있는지를 고찰해 보았다. 연구 결과, 같은 용어에 대한 두 대상의 의미상의 차이와 두 차원으로 구성되고 반복 진행되는 RUP의 특수성 때문에, 객체지향 환경에서 기반 표준으로써 ISO/IEC 12207을 적용하는데는 약간의 문제점이 발생함을 알 수 있었다. 뿐만 아니라 이러한 문제점을 보안하기 위한 방법을 모색해 보았다. ISO/IEC 12207에서 제시하는 항목들의 크기(granularity)가 크고 개괄적이어서 특정 프로세스에 적용하기가 매우 막연하기 때문에, 본 논문에서 제시한 내용을 바탕으로 객체지향 환경에서 사용 가능하도록 표준을 실제 세분화하는 연구가 향후 필요하다.

참고문헌

[1] Jim Moore, "ISO 12207 and Related Software Life Cycle Standards", http://www.acm.org/tsc/lifecycle.html

[2] "Heritage of Systems Engineering Standards" http://www.incose.org/stc/standards-evolution.htm

[3] Lewis Gray, "ISO/IEC 12207 Software Life Cycle Processes", http://www.stsc.hill.af.mil/crosstalk/1996/aug/isoiec.asp

[4] ISO, "ISO/IEC 12207 International Standard, Information Technology - Software Life Cycle Processes", Aug, 1995

[5] Rational Software Corporation Whitepaper, "Rational Unified Process - Best Practives for Software Development Teams"

[6] Rational Software Corporation Whitepaper, "A Rational Approach to Software Development Using Rational Rose 4.0 http://www.rational.com/sitewide/support/whitepapers/

[7] Ivar Javobson, Grady Booch, James Rumbaugh, "The Unified Software Development Process", Addison Wesley Longman, Inc., Jan. 1999

[8] Philippe Kruchten, "The Rational Unified Process", Addison Wesley Longman, Inc., Nov. 1998

[9] Philippe Kruchten,"Architectural Blueprints - The "4+1" view model of software architecture", IEEE Software 12, 6 (November 1995), 42-50

[10] IEEE, "Draft Recommended Practice for Architectural Description IEEE P1471/D5.2, Dec, 1999