DOI: 10.3745/KIPSTD.2010.17D.6.443

COTS 시스템 기반 속성 및 행위 분석에 의한 생명주기에 관한 연구

이 은 서[†] · 김 중 수[††]

요 약

COTS(Commercial Off The Shelf) 시스템은 점점 더 재사용 개발에 있어서 중요한 일중의 하나로 되어가고 있다. 그것은 개발될 소프트웨어 품질 혹은 프로젝트 일정에 영향을 미칠 수 있는 COTS의 신뢰성을 포함한다. COTS 분석의 결과가 생길 수 있는 위험의 결과와 함께 프로젝트에 문서화되어야 한다. 효율적인 위험관리는 문제에 쉽게 대처할 수 있게 해주며, 그것이 수용할 수 없는 예산이나 일정 지연이 되지 않도록 해준다. 본 연구에서는 COTS 개발 시, 프로세스 이정표와 노력에 관한 위험요소 분석에 대한 기준을 제시한다.

키워드 : 위험관리, COTS(Commercial Off The Shelf), 결함관리, 생명주기, 속성분석, 행위분석

A Study of Analysis of Attribute and Operation based on COTS System

Lee Eun Ser[†] · Kim Joong Soo[††]

ABSTRACT

COTS system is increasingly seen as one of the main jobs of reuse development. It involves reliability of COTS that might affect the project schedule or the quality of the software being developed and taking action to avoid these risks. The results of the COTS analysis should be documented in the project plan along with an analysis of the attribute and operation. Effective risk management makes it easier to cope with problems and to ensure that these do not lead to unacceptable budget or schedule slippage. This research provides criteria of analysis of risk items to the estimation of process milestone on COTS development.

Keywords : Risk Management, COTS(Commercial Off The Shelf), Defect Management, Life Cycle, Attribute Analysis, Operation Analysis

1. 서 론

위험 관리는 점점 더 프로젝트 관리자에게 있어서 중요한 일중의 하나로 되어가고 있다. 그것은 개발될 소프트웨어의 품질 혹은 프로젝트 일정에 영향을 미칠 수 있는 위험을 예측하는 것을 포함한다[1, 6, 7]. 위험 분석의 결과가 생길 수 있는 위험의 결과와 함께 프로젝트 계획에 문서화되어야 한다. 효율적인 위험 관리는 문제에 직면했을 때 쉽게 해결할 수 있으며, 그것을 수용할 수 없는 예산이나 일정 지연이 되지 않도록 해준다.

위험을 일어나지 않도록 하는 것으로 생각할 수 있으며, 위험은 프로젝트, 개발될 소프트웨어, 혹은 조직을 위협한다[2, 8, 9].

위험은 소프트웨어가 개발되는 환경과 조직, 프로젝트에 따라서 프로젝트에 영향을 미친다.

위험 관리는 대부분의 프로젝트가 직면하고 있는 본질적인 불확실성 때문에 소프트웨어 프로젝트에서 특히 중요하다. 이러한 것은 느슨하게 정의된 요구사항과 소프트웨어 개발, 각 개인의 기술차이, 고객의 요구 변경으로 인한 요구사항 변경에 필요한 시간과 자원을 평가하는 데에서 오는 어려움에 기인한다. 위험을 예측해야 하고, 프로젝트, 제품, 비즈니스에서 이러한 위험의 영향을 이해해야 하고, 이러한 위험을 피할 수 있는 단계를 취해야 한다[3, 10]. 따라서 본 논문에서는 COTS 재사용 시 발생할 수 있는 위험요소를

† 종신회원 : 안동대학교 컴퓨터공학과 조교수
†† 정 회 원 : 안동대학교 컴퓨터공학과 교수(교신저자)
논문접수 : 2010년 9월 2일
수 정 일 : 1차 2010년 9월 13일, 2차 2010년 9월 14일
심사완료 : 2010년 9월 14일

속성과 행위 관점에서 분석하고자 한다. 또한 점증적으로 위험요소를 줄이기 위한 생명주기를 제시하고 각 단계별 개발과정을 기반으로 하여 위험관를 수행하고자 한다. 그리고 산출된 결과를 기반으로 하여 소프트웨어 성숙 색인에 의하여 정량화하여 신뢰성을 확인하고자 한다.

2. 기반연구

2.1 나선형 모형

USC(미국 남가주대학교)의 보헴은 소프트웨어 개발 프로세스의 위험관리 측면에서 나선형 모형을 제안하였다. 나선형 모형은 반복적인 개발방법과 유사하다. 중요한 특징은 개발을 위한 계획 및 요구분석 후에 위험요소와 차선책에 대하여 검토하는 단계가 있다는 점이다. 즉 프로젝트의 초기에 실패 요인과 위험 요소들을 찾아내어 대비하자는 것이다[1, 11].

나선형 모형은 반복적으로 앞으로 나가는 나사 모양이 된다. 계획, 설계, 개발, 평가의 개발 주기가 한 번에 그치는 것이 아니라 시스템을 여러 부분으로 나누어 여러 번의 개발 주기를 거치면서 시스템이 완성된다. 이론적으로는 외부 나선 주기가 위험이 적은데 그 이유는 반복 과정을 거치면서 시스템이 더욱 구체화되고 이해도가 높아지기 때문이다. 시스템을 이루는 여러 기능 중에서 성패를 좌우할 만한 기능들을 먼저 개발하고 다른 기능들을 후에 추가해 나가는 점증적인 방법이다.

2.2 소프트웨어 성숙 색인

소프트웨어 성숙 색인(Software Maturity Index, SMI) 새로운 소프트웨어 개발과 존재하는 소프트웨어의 유지보수를 위해 사용할 수 있다[4, 12, 13].

IEEE Std. 982.1-1998은 제품의 각 릴리즈에서 발생하는 변경 사항에 기초해서 소프트웨어 제품의 안전성에 관한 지표를 제공하는 소프트웨어 성숙 색인을 제안했다[5].

소프트웨어 성숙 색인을 산출하기 위해서는 다음의 정보가 필요하다.

M_T = 현재 릴리즈에서 모듈의 수
F_c = 현재 릴리즈에서 변경된 모듈의 수
F_a = 현재 릴리즈에서 추가된 모듈의 수
F_d = 현재 릴리즈에서 삭제된 이전 릴리즈의 모듈의 수

소프트웨어 성숙 색인은 다음 방법으로 계산된다.

$$SMI = [M_T - (F_a + F_c + F_d)]/M_T$$

SMI가 1.0에 가까워지면 제품은 안정되기 시작한다. 또한 SMI는 소프트웨어 유지보수 활동을 계획하는 척도로 사용된다. 소프트웨어 제품의 릴리즈를 생산하는 평균 시간은 SMI와 상호 연관시킬 수 있다.

2.2 문제점 제기

나선형 모형에서는 계획수립, 위험분석, 개발, 평가의 네 단계로 구성되어 있다. 나선형 모형은 위험요소를 찾아서 제거하는 것이 목적이며, 이를 위하여 점증적인 단계로 접근을 하게 되어 있다.

그러나 찾으려는 위험요소는 개발단계의 관리적인 측면과 과정에 중점을 두고 있어서 개발의 상세한 위험요소를 찾는 데는 한계가 있다. 따라서 본 논문에서는 COTS 재사용시 활용할 수 있는 생명주기를 제시한다. 제안하고자 하는 생명주기는 속성과 행위를 분석해서 위험요소를 줄여서 전체적인 시스템의 신뢰성을 향상시키고자 한다.

3. 본 론

소프트웨어 개발 방법론과 생명주기는 개발을 효과적 및 효율적으로 관리하기 위하여 필요한 필수요소가 되고 있다. 생명주기의 경우 프로젝트의 진척도와 신뢰성까지 영향을 미치므로 간과 될 수 없는 요소가 되고 있다.

그러나 개발되는 영역과 범위가 너무 가변적이어서 모든 영역을 포함할 수 있는 범용적인 생명주기는 존재하기 어렵다. 따라서 생명주기의 적용 방법을 프로젝트의 특성을 고려하는 것과 개발하는 형식에 의한 방법으로 접근 할 수 있다.

본 논문에서는 재사용성을 고려한 개발 방법 중 COTS (Commercial off-the-shelf 이하 COTS로 표기)를 대상으로 하는 경우에 오류를 줄일 수 있는 효율적인 생명주기를 제안하고자 한다. COTS용 생명주기를 완성하기 위하여 COTS 개발시 영향을 줄 수 있는 요소를 추출해야 하며, 이를 기반으로 하여 생명주기를 완성하고자 한다.

3.1 COTS의 문제점 추출 (incremental or spiral)

COTS를 위한 생명주기 설계를 위하여 선행되어야 할 것은 COTS 사용시 발생되는 문제점들을 추출하는 것이다. 이와 같은 문제점 추출은 COTS 지원 생명주기 설계시 고려되어야 할 중요한 사항이며, 생명주기의 핵심 요소가 된다. COTS 사용시 발생하는 문제는 다음과 같이 세 가지 관점에서 발생할 수 있다.

3.1.1 COTS 기능이나 성능의 제어

COTS의 재사용 관점에서 발생할 수 있는 문제점에 대하여 다음과 같은 세부사항들이 발생될 수 있다.

- COTS 통합에서폭포수모형사용 : 진척도 관리를 위하여 폭포수 모형에 의한 생명주기 진행시에 발생되는 문제점이다.

- COTS를 재사용한 전개 개발과정에서 변경 사항이 많아진다.
- COTS 성능에 대한 신뢰감 : COTS의 재사용성을 활용하여 개발이 진행되는 과정에서 변경사항과 같은 변동성에 의한 성능저하 문제가 발생한다.

3.1.2 COTS 상호간의 호환성 결여

- COTS 규격을 확인하지 않은 결합에 대한 선급한 효과 기대 : COTS의 개발시 규격이 일치하지 않아서 문제가 발생한다.
- 규격에 맞지 않고 너무 많은 COTS 통합단계에 문제가 발생한다.
- 재사용할 COTS의 개발이 완성될 때 까지 통합을 연기해야 됨으로써 전체적인 개발주기에 영향을 미치게 된다.

3.1.3 COTS 전개에 대한 제어

- COTS 기능관점의 아키텍처 문제에 의하여 개발 관점의 표준화가 이루어지지 않는다.
- 여러 규격에 의한 소프트웨어 유지보수시에 기준마련이 어렵다.
- COTS는 독립적으로 존재하지만 엄격한 결합이 필요하다.
- 유지보수와 같은 COTS 전개공정에 대한 제어의 어려움

3.2 COTS 위한 해결방안

3.1절에서 제시한 문제점을 해결하기 위하여 해결책을 제시하고자 한다. 제시된 해결책은 COTS 사용시 발생되는 문제점을 해결 및 보완할 수 있는 생명주기 설계시 활용되게 된다.

3.2.1 COTS 기능이나 성능의 제어

구현하려는 COTS 기능과 부합되지 않거나 일관성 있는 성능을 유지하지 못하는 경우에 발생하게 된다. 이와 관련된 문제점과 해결책은 <표 1>과 같다.

3.2.2 COTS 상호간의 호환성 결여

재사용성을 위한 COTS 설계시 상호연관성 및 호환성과 관련되어 문제가 발생되는 경우이다. 이와 관련된 문제점과 해결책은 <표 2>와 같다.

3.2.3 COTS 전개에 대한 제어

독립적으로 실행되는 COTS가 개발의 모든 과정에서 제어가 되지 않아서 발생되는 경우이다. 이와 같은 문제는 COTS가 개별적인 형태로는 문제가 없지만 통합과정에서 발생되는 문제이다. 이와 관련된 문제점과 해결책은 <표 3>과 같이 제시하였다.

〈표 1〉 COTS 기능이나 성능의 제어에 대한 문제점과 해결책

문제점	해결책
□ COTS 통합에서폭포수모형사용 : 진척도 관리를 위하여 폭포수 모형에 의한 생명주기 진행시에 발생되는 문제점이다.	□ 위험분석과 진척도 관리를 위하여 나선형 구조모델과 폭포수 모델의 마일스톤을 활용한다.
□ COTS를 재사용한 전개 개발과정에서 변경 사항이 많아진다.	□ COTS의 재사용부분에서 가변성에 해당하는 부분을 추출한다.
□ COTS 성능에 대한 신뢰감 : COTS의 재사용성을 활용하여 개발이 진행되는 과정에서 변경사항과 같은 변동성에 의한 성능저하 문제가 발생한다.	□ 신뢰성 향상을 위하여 추적성을 활용한다.

〈표 2〉 COTS 상호간의 호환성 결여에 대한 문제점과 해결책

문제점	해결책
□ COTS 규격을 확인하지 않은 결합에 대한 선급한 효과 기대 : COTS의 개발시 규격이 일치하지 않아서 문제가 발생한다.	□ COTS 기능 구현시 요구사항대비 추적성을 유지하여 개발 규격의 일관성을 유지한다.
□ 규격에 맞지 않고 너무 많은 COTS 통합단계에 문제가 발생한다.	□ COTS 통합시 오류 발생을 줄이기 위한 공용성과 가변성을 구분하고 계층화한다.
□ 재사용할 COTS의 개발이 완성될 때 까지 통합을 연기해야 됨으로써 전체적인 개발주기에 영향을 미치게 된다.	□ 개발단계에 마일스톤을 설정하여 단위별 검증을 하여 전체 개발주기에서 오류를 최소화하여 일정에 차질이 없도록 한다.

〈표 3〉 COTS 전개에 대한 제어에 대한 문제점과 해결책

문제점	해결책
□ COTS 기능관점의 아키텍처 문제에 의하여 개발 관점의 표준화가 이루어지지 않는다.	□ 최근 동향의 표준을 참조한다.
□ 여러 규격에 의한 소프트웨어 유지보수시에 기준마련이 어렵다.	□ 유지 보수를 원활히 하기 위하여 기능별 개발 속성과 오퍼레이션을 구분한다.
□ COTS는 독립적으로 존재하지만 엄격한 결합이 필요하다.	□ 결합의 신뢰성을 향상시키기 위하여 오류검출 후 개선사항을 적용하여 점층적으로 개발한다.
□ 유지보수와 같은 COTS 전개공정에 대한 제어의 어려움	□ COTS 전개공정에 대한 제어를 위하여 COTS용 생명주기를 만들고, 이를 기반으로 제어한다.

3.3 COTS 재사용시 문제점 해결을 방법

COTS의 문제점을 예방하기 위하여 이를 분석하였으며, 이에 대한 대안을 3.1과 3.2절에서 제시하였다. 본 절에서는 각각의 문제점을 예방하고 발생시 해결하기 위한 작업의 단계를 생명주기에 의하여 해결하고자 한다. 제안한 생명주기는 COTS의 재사용시 공용성과 가변성을 기능관점에서 구분하기 위함이다. 이와 같은 과정을 반복적 수행하여 위험을 확인하게 되고 점층적으로 완성도를 높여서 COTS의 재사용시 신뢰성을 향상시키고자 한다.

3.1과 3.2절에 제시하였던 문제점과 해결책을 생명주기에 반영하고자 한다. <표 4>, <표 5>, <표 6>은 문제점과 해결책을 반영하기 위한 추적성 표이다.

3.4 COTS 재사용을 위한 생명주기 설계

3.1에서부터 3.3절까지 제시한 내용을 기초로 하여 COTS의 재사용을 위한 생명주기를 설계하고자 한다.

제안하는 생명주기는 4단계의 구조를 갖는다. 초기화 단계, 문제점 분석 단계, 문제점 확인 단계, 개선 및 적용단계이다.

각 단계는 위험요소를 찾고 제거하기 위하여 나선형 구조

3.3.1 COTS 기능이나 성능의 제어

〈표 4〉 COTS 기능이나 성능의 제어에 대한 추적성 표

문제점	해결책	생명주기
□ COTS 통합에서폭포수모형사용 : 진척도 관리를 위하여 폭포수 모형에 의한 생명주기 진행시에 발생되는 문제점이다.	□ 위험분석과 진척도 관리를 위하여 나선형 구조모델과 폭포수 모델의 마일스톤을 활용한다.	□ Cycle에 의하여 진척도를 확인한다.
□ COTS를 재사용한 전개 개발과정에서 변경 사항이 많아진다.	□ COTS의 재사용부분에서 가변성에 해당하는 부분을 추출한다.	□ 생명주기에서 분석과정과 추적성을 통하여 개선된 내용을 반영하도록 한다.
□ COTS 성능에 대한 신뢰감 : COTS의 재사용성을 활용하여 개발이 진행되는 과정에서 변경사항과 같은 변동성에 의한 성능저하 문제가 발생한다.	□ 신뢰성 향상을 위하여 추적성을 활용한다.	□ 요구사항대비 속성과 행위의 정확성을 확인하여 지원한다.

3.3.2 COTS 상호간의 호환성 결여

〈표 5〉 COTS 상호간의 호환성 결여에 대한 추적성 표

문제점	해결책	생명주기
□ COTS 규격을 확인하지 않은 결합에 대한 선급한 효과 기대 : COTS의 개발시 규격이 일치하지 않아서 문제가 발생한다.	□ COTS 기능 구현시 요구사항대비 추적성을 유지하여 개발 규격의 일관성을 유지한다.	□ 추적성과 계층구조에 적합하게 기능화하여 규격을 통일화한다.
□ 규격에 맞지 않고 너무 많은 COTS 통합단계에 문제가 발생한다.	□ COTS 통합시 오류 발생을 줄이기 위한 공용성과 가변성을 구분하고 계층화한다.	□ 기능을 속성과 행위 관점으로 분석하여 공용성과 가변성을 구분한다.
□ 재사용할 COTS의 개발이 완성될 때 까지 통합을 연기해야 됨으로써 전체적인 개발주기에 영향을 미치게 된다.	□ 개발단계에 마일스톤을 설정하여 단위별 검증을 하여 전체 개발주기에서 오류를 최소화하여 일정에 차질이 없도록 한다.	□ Cycle에 의하여 진척도를 확인한다.

3.3.3 COTS 전개에 대한 제어

〈표 6〉 COTS 전개에 대한 제어에 대한 추적성 표

문제점	해결책	생명주기
□ COTS 기능관점의 아키텍쳐 문제에 의하여 개발 관점의 표준화가 이루어지지 않는다.	□ 최근 동향의 표준을 참조한다.	□ 계층구조 기능화를 통하아키텍쳐 설계를 완성한다.
□ 여러 규격에 의한 소프트웨어 유지보수시에 기준마련이 어렵다.	□ 유지 보수를 원활히 하기 위하여 기능별 개발 속성과 오퍼레이션을 구분한다.	□ 점층적으로 속성과 행위가 정확한지 확인한다.
□ COTS는 독립적으로 존재하지만 엄격한 결합이 필요하다.	□ 결합의 신뢰성을 향상시키기 위하여 오류검출 후 개선사항을 적용하여 점층적으로 개발한다.	□ 개선사항을 적용하기 위하여 점층적으로 개선된 속성과 행위를 계층구조에 적용한다.
□ 유지보수와 같은 COTS 전개공정에 대한 제어의 어려움	□ COTS 전개공정에 대한 제어를 위하여 COTS용 생명주기를 만들고, 이를 기반으로 제어한다.	□ Cycle은 네 단계로 구분하여 제어한다.

로 구성을 하였다. 그리고 각 단계별로 주기가 증가될수록 점층적인 방법을 도입하여 개발 내용의 성숙도를 높이고자 하였다.

점층적인 방법으로 반복적인 작업이 수행되기 때문에 진척도 관리가 필요하다. 따라서 cycle을 네 단계로 구분하여 진척도 관리를 수행하고자 한다.

3.4.1 초기화 단계

초기화 단계에서는 각 단계의 기초적인 분석 작업을 수행하게 된다. 대상은 요구사항과 속성, 행위 및 계층구조화이다. 초기화 단계에서 수행되는 작업은 다음과 같다.

① 요구사항 분석 : 기능을 구현하기 위한 요구사항 분석이 수행된다.
② 요구사항의 속성 분석 : 요구되는 기능이 시스템으로 완성되기 위해서는 속성 분석이 수행되어야 한다. 속성 분석은 기능이 가지고 있는 속성을 일반화 하는 과정이다.
③ 기능별 행위 분석 : 분석된 속성을 시스템 상에서 실행시키기 위하여 행위를 분석해야 한다. 분석된 행위는 실제로 함수 또는 메소드의 형태로 만들어지게 된다.
④ 계층구조화를 위한 요구사항 분석(공용성, 가변성) : 기능을 구조화하기 위하여 공통적으로 사용되는 기능과 변화가능성이 있는 기능을 구분하여 속성과 행위를 공용성과 가변성으로 구분하게 된다. 구분된 속성과 행위는 계층화하여 재사용성을 높이게 된다.

3.4.2 문제점 분석 단계

초기화 단계에서는 하나의 요구사항을 기능화하기 위한 속성과 행위 분석에 중점을 두었다. 문제점 분석 단계는 독립적인 기능에서는 문제가 발생되지 않지만, 다른 기능과 연동되는 과정에서 발생하는 문제를 예방하기 위한 단계이다. 따라서 분석되는 기준은 초기화 단계와 같이 요구사항과 속성, 행위 및 계층구조화이지만, 서로의 연동되는 과정에서 발생되는 문제의 원인에 중점을 두고 있다. 초기화 단계의 성숙도를 높이기 위한 고도화 단계이다. 문제점 분석 단계에서 수행되는 작업은 다음과 같다.

① 요구사항간의 연관성 분석 : 초기화 단계에서 분석된 요구사항간의 연관성을 입력과 출력 관점에서 분석하여 연관관계를 정의한다.
② 분석된 요구사항간의 속성 연관성 분석 : 초기화 단계에서 분석된 속성을 요구사항간의 연관성 분석 작업에 의하여 속성간의 연관성을 분석한다.
③ 행위의 연관성 분석 : 초기화 단계에서 분석된 행위를 요구사항간의 연관성 분석 작업에 의하여 행위간의 연관성을 분석한다.
④ 계층구조의 기능화 : 초기화 단계에서 분석된 행위를 요구사항간의 연관성 분석 작업에 의하여 계층구조를 기능에 의하여 분석한다.

3.4.3 문제점 확인 단계

초기화 단계에서는 요구사항을 독립적인 기능으로 만들고, 문제점 분석단계에서는 각 기능화하는 과정에서 다른 요구사항간의 연관관계를 기초로 분석을 하게 된다. 문제점 확인 단계에서는 앞의 두 단계에서 파악된 문제의 원인을 수정하고 개선하기 위한 기초단계가 된다. 따라서 문제점 원인을 올바르게 수정하여 다른 기능에 오류가 전이되는 것을 방지하고자 한다. 문제점 분석 단계에서 수행되는 작업은 다음과 같다.

① 요구사항 정의 : 오류를 내포하고 있는 요구사항을 수정한다.
② 요구사항 대비 속성의 정확성 확인 : 오류를 내포하고 있는 속성을 수정한다.
③ 요구사항 대비 행위의 정확성 확인 : 오류를 내포하고 있는 행위를 수정한다.
④ 계층구조 기능의 확인 : 오류를 내포하고 있는 요구사항, 속성 및 행위를 계층구조의 기능중심으로 수정한다.

3.4.4 개선 및 적용단계

초기화, 문제점 분석, 문제점 확인 단계를 기반으로 개선된 요구사항을 개선하여 적용하기 위한 단계이다.

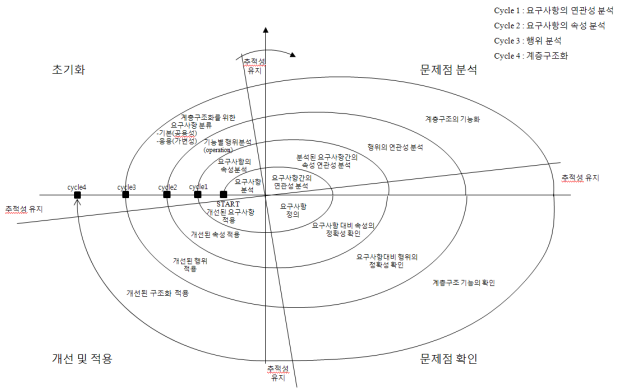
① 개선된 요구사항 적용 : 이전 단계에서 분석된 요구사항 오류를 개선하여 적용한다.
② 개선된 속성 적용 : 이전 단계에서 분석된 속성 오류를 개선하여 적용한다.
③ 개선된 행위 적용 : 이전 단계에서 분석된 행위 오류를 개선하여 적용한다.
④ 개선된 구조화 적용 : 이전 단계에서 분석된 계층구조화의 기능 오류를 개선하여 적용한다.

3.4.5 공통 사항

앞에서 제시한 네 단계는 요구사항을 정확히 기능화하기 위하여 수행되는 단계이다. 따라서 요구사항이 계층구조화로 전이되는 단계에서 정확히 분석되고 이를 확인하기 위하여 추적성이 필요하게 된다. 추적성은 네 단계에 모두 적용이 되어서 요구사항, 속성, 행위, 계층구조의 기능화의 변경 이력사항을 유지하게 된다. 결과적으로 개선을 위한 유지보수의 효율성을 높이기 위함이다.

위의 사항을 기반으로 하여 생명주기를 설계하고자 한다. 생명주기는 네 단계의 세부사항을 포함하고 있으며, cycle을 기반으로 진척도 측정을 수행하게 된다. 제안된 생명주기는 (그림 1)과 같다.

각 cycle은 점층적으로 해당 작업이 완성되게 된다. cycle1은 요구사항의 연관성 분석을 완료하게 되며, cycle2



(그림 1) COTS 재사용을 위한 생명주기

는 요구사항의 속성 분석을 완성하는 이정표가 된다. cycle3는 행위분석과 cycle4는 개선된 사항을 적용하는 계층구조화의 이정표가 된다. 따라서 cycle에 의하여 진척도와 성숙도를 확인할 수 있게 된다.

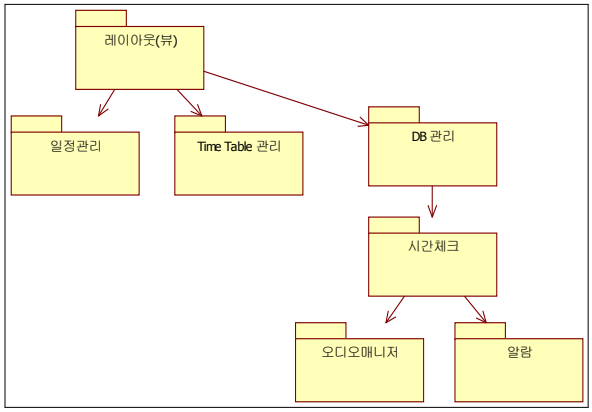
4. 사례연구 및 검증

본 연구에서는 제시한 생명주기를 적용하여 확인하기 위하여 사례연구를 수행하였다. 또한 소프트웨어 성숙 색인을 이용하여 정량화 비교를 수행하였다. 결과물에서 제시되는 내용은 설계에 중점을 두었으며, 실험은 다음과 같은 환경에서 수행하였다.

4.1 환경정의 및 요구사항 추출

스마트 폰에서의 시간표 기능을 보완 및 개발하여 일정관리뿐만 아니라 일정에 따른 휴대전화의 매너모드를 관리하는 것이다. 사용자의 실수로 인한 피해를 줄일 수 있는 상황을 줄임으로써 상대방에 대한 에티켓의 수준을 높이고, 휴대전화의 사용자가 매너모드에 관해서 자동제어를 적용하고자 한다.

4.1.1 도메인 정의



도메인	설명
오디오 매니저	-휴대폰의 벨 설정을 관리 -휴대폰 알람의 벨소리를 관리 -사용자가 설정한 상태로 벨모드, 진동모드, 무음모드로 변경이 가능 -기본적으로 현재 설정에 따라, 현재 설정이 벨 모드이면 진동모드로 변경되고, 진동모드이면 벨 모드로 변경 -알람 모드는 설정에 따라 진동모드 관계없이 알람, 진동모드 일 경우 진동 등 사용자 설정이 가능
시간 체크	-시작 시간과 종료 시간 두 가지 시간을 입력받음 -시작 시간이 되면 오디오 매니저를 통해 진동모드로 변경 -종료 시간이 되면 기존의 모드로 다시 돌아옴 -시간이 설정 되었을 때 설정 시간이 현재 시간에 포함되는지 체크하기 위해 시작시간, 종료시간, 설정 하는 시간을 기준으로 시간을 체크
DB 관리	-설정된 시간과 메모(일정)을 저장 -인터페이스를 통해 사용자가 일정 등을 입력 및 수정할 수 있음 -또한 저장된 값이 필요할 때 값을 시간체크에서 불러와 사용할 수 있음
뷰	-일정관리, 알람 표시, 시간표 관리 등의 사용자 인터페이스를 보여줌 -알람이 실행될 때는 Toast를 통하여 소리와 함께 알람 현황을 알려줌 -사용자의 필요에 따라 일정관리, 시간표 관리를 선택적으로 표시
타임 테이블	-[월~금요일][9시~18]의 일과시간 기준으로 타임테이블 인터페이스를 구성하여 간단히 버튼 한번으로 1시간단위 일정관리가 가능 -원하는 시간대의 버튼을 눌러서 메모를 입력 할 수 있고 등록된 부분은 버튼에 등록되었다고 표시 -또한 이미 설정된 시간대를 다시 수정 또는 삭제 할 수 있음 -등록된 일정은 DB관리를 통해 DB에 저장됨
일정	-일과 시간 외의 시간을 매너모드나 알람을 설정할 수 있음 -시작시간과 종료시간을 직접 설정 할 수 있음 -분 단위로 시간 체크를 요청 함
알람	-벨 관리와 독립적으로 설정된 알람시간에 따라서 오디오 매니저를 통하여 오디오 매니저 설정에 따라 알람을 수행

4.1.2 기능적 요구조건

- 매너 모드와 벨소리모드의 전환기능
- 정해진 시간에 벨소리에서 매너 모드로 전환기능
- 지정된 시간 정보와 DB연동
- 시간표 설정하는 UI에 시간설정이 되면 그 시간에는 매너모드로 전환
- 스케줄 이외의 시간에는 기존의 모드(벨소리/매너모드)로 자동 전환

－스케줄에 맞는 DB테이블 설계 및 연동
－자신의 스케줄을 다른 상대방과 공유(전송) 기능
－스케줄에 대한 예약 알람 선택 기능

4.1.3 비기능적 요구조건

① 보안성 요구사항
－타인이 사용자의 스케쥴, 일정 등을 보는 것을 방지하기 위해 Password 인증 으로 프로그램을 실행하고, 데이터를 입력 또는 삭제한다.
－Password 인증을 사용자가 사용 또는 비사용으로 선택할 수 있다.

② 신뢰성 요구사항
－데이터의 접근 신뢰도와 수행 기능은 전적으로 Android Platform에 일임한다.
－모든 부분에 대한 추적성이 유지되어 시스템의 신뢰도를 높인다.

③ 사용성 요구사항
－사용자의 일정을 보다 쉽게 등록 할 수 있도록 하며, 스마트폰 특성에 맞춘 터치패드의 활용을 극대화하도록 한다.

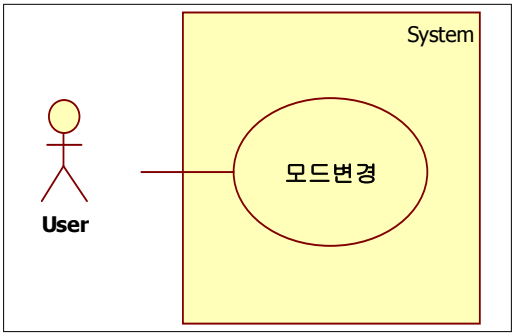
④ 유지보수성 요구사항
－앱스 본연의 특성에 맞게 기존의 시스템의 업그레이드가 용이하도록 가변성, 공용성을 고려하여 설계, 개발한다.

4.2 COTS 재사용을 위한 생명주기 적용 결과물

제안한 COTS 재사용을 위한 생명주기의 적용에 의하여 각 사이클마다 결과를 제시하였다.

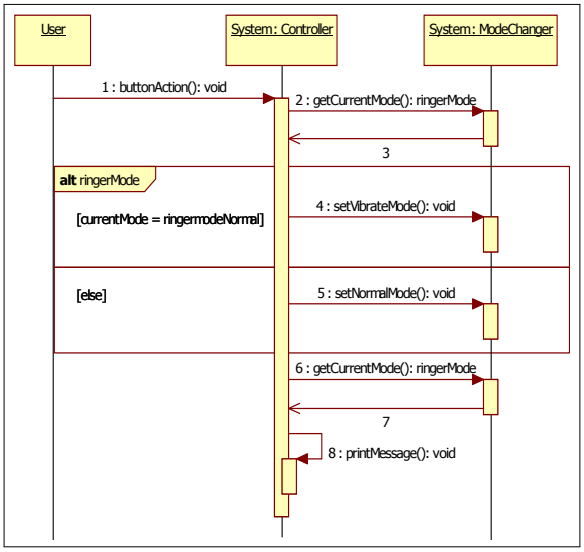
4.2.1 사이클 1

① 유스케이스 다이어그램

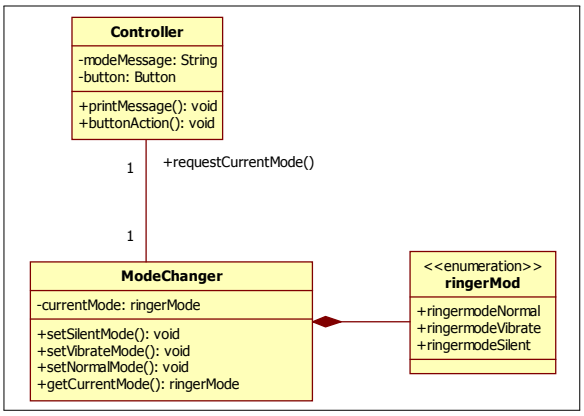


◎ 액터 : 시간 관리자
◎ 유스케이스 : 모드변경
◎ Communication : 액터와 모드변경
◎ 시스템바운더리 : MMA v0.1

② 시퀀스 다이어그램

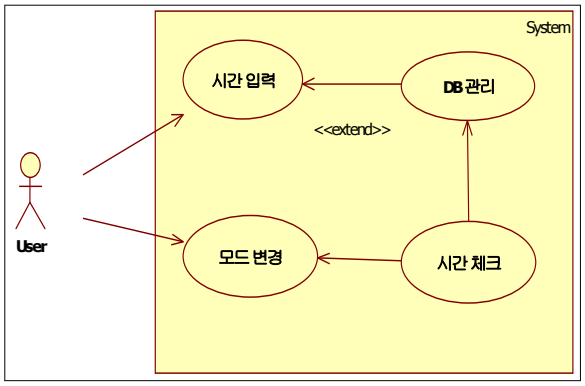


③ 클래스 다이어그램



4.2.2 사이클 2

① 유스케이스 다이어그램

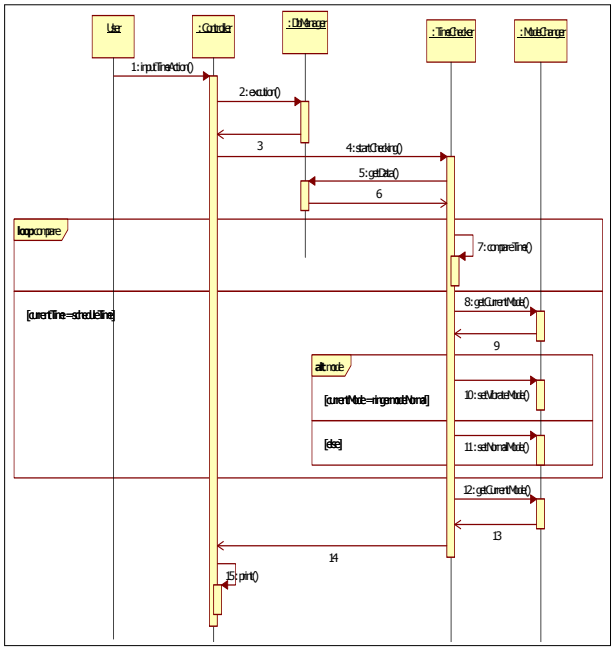


◎ 액터 : 사용자
◎ 유스케이스 : 모드변경, 시간입력, DB관리, 시간체크

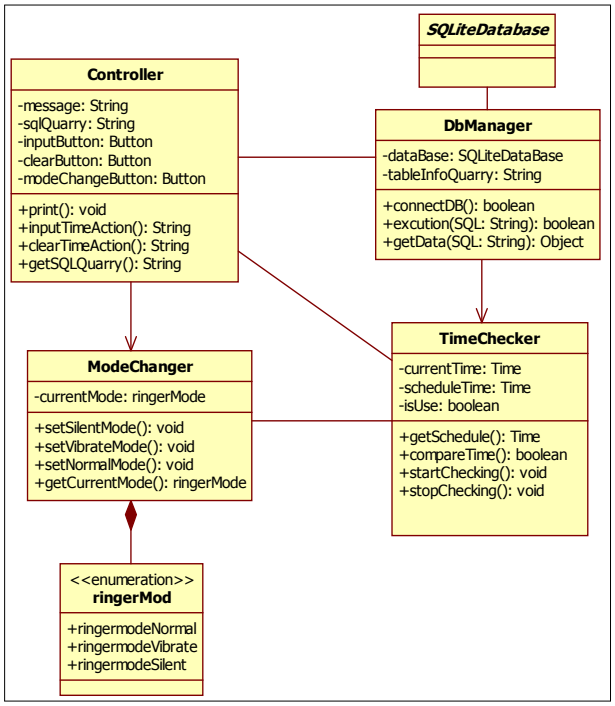
◎ Communication : 액터와 모드변경, 액터와 시간입력, DB관리와 시간입력, 시간체크와 DB관리, 시간체크와 모드변경

◎ 시스템바운더리 : MMA v0.2

② 시퀀스 다이어그램

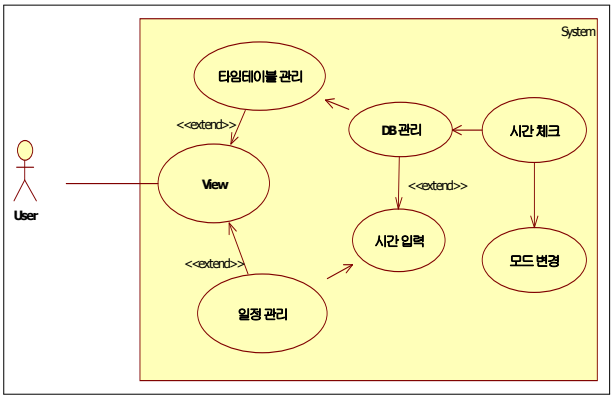


③ 클래스 다이어그램



4.2.3 사이클 3

① 유스케이스 다이어그램



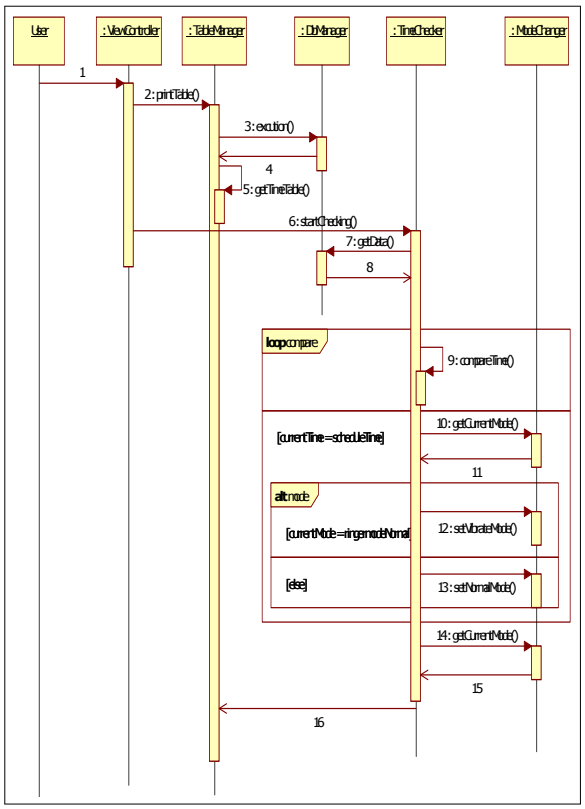
◎ 액터 : 사용자

◎ 유스케이스 : 모드변경, DB관리, 시간체크, 일정관리, 타임테이블관리, 시간입력, View

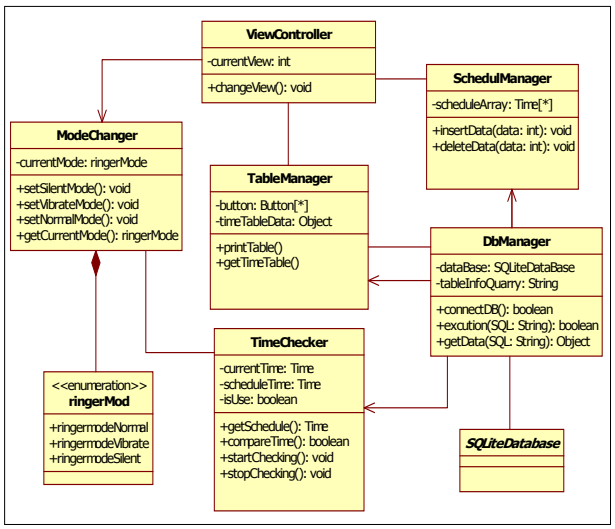
◎ Communication : User와 View, View와 타임테이블관리, View과 일정관리, DB관리와 타임테이블관리, DB관리와 시간입력, DB관리와 시간체크, 시간체크와 모드변경

◎ 시스템바운더리 : MMA v0.3

② 시퀀스 다이어그램



③ 클래스 다이어그램



4.3 소프트웨어 성숙 색인에 의한 정량화 비교

제안된 COTS 재사용을 위한 생명주기의 각 사이클에 의한 결과물은 4.1과 4.2절에서 제시하였다. 제시된 결과물에서는 위험요소를 제거하고 고도화를 측정하기 위하여 소프트웨어 성숙 색인을 활용하였다. 소프트웨어 성숙 색인은 2장에서 설명하였으며, 그 산술식은 다음과 같다.

$$SMI = [M_T - (F_a + F_c + F_d)]/M_T$$

전체 사이클은 3개이지만 사이클 1은 처음 생성된 결과물이므로 모듈의 변경 및 추가, 삭제를 측정할 수 가 없다. 따라서 사이클 1을 기준으로 사이클 2를 비교하였고, 사이클 3은 사이클 2를 기준으로 산정하였다.

소프트웨어 성숙 색인에 의하여 사이클 3이 사이클 2보다 향상된 것을 확인할 수 있다.

〈표 7〉 COTS 재사용을 위한 생명주기의 소프트웨어 성숙 색인

	M_T	F_a	F_c	F_d	SMI
사이클 2	6	3	1	0	0.33
사이클 3	8	2	1	0	0.625

5. 결 론

본 논문에서는 COTS 재사용시 발생하는 결과물의 신뢰성을 높이고자 생명주기를 제시하였다. 생명주기는 사이클을 세분화하여 진척도 측정과 단계별로 속성 및 행위의 성숙도를 향상시키고자 했다. 또한 제안된 생명주기의 효과를 확인하기 위하여 소프트웨어 성숙 색인을 확인하였으며, 그 결과 0.33에서 0.625라는 수치로 많은 수준의 향상을 보였다. 산출된 경과에 의하여 제안된 COTS 재사용 생명주기의 효용성을 확인하였다.

향후 연구로는 현재 제시된 정량화 식을 자동으로 산출할 수 있는 도구를 개발하는 것이다. 또한 산출식의 정확도를 검증하여 신뢰도를 높이고 이를 위한 보정작업과 함께 여러 과제에 적용하는 과정이 요구되며 이를 계획하고 있다.

참 고 문 헌

[1] 최은만, 소프트웨어공학론, 사이텍미디어, 2001.
[2] 윤청, 성공적인 소프트웨어 개발 방법론, 생능출판사, 1999.
[3] 권기태, 남영광, 소프트웨어공학, 홍릉과학출판사, 2008. 02.
[4] 우치수, 소프트웨어공학 실무적 접근, 한산출판사, 2003.
[5] Software Engineering Standards, IEEE 1994 edition, 1994
[6] Robert h. dunn, “Software defect removal”, Mcgraw-hill, 1984.
[7] Ram chillarrege, Kothanda r. prasad, “Test and development process retrospective a case study using ODC triggers”, IEEE computer society, 2002.
[8] Wohlin, Runeson, “Defect content estimations from review data”, Proceedings international conference on software engineering ICSE pp.400-409, 1998.
[9] Gaffney, John, “Some models for software defect analysis”, Lockheed martin, 1996.
[10] B. compton and C. withrow, “Prediction and control of ada software defects”, J. systems and software, Vol.12, pp. 199-207, 1990.
[11] Roger s. pressman “Software engineering” Mcgraw-hill international edition, 1997.
[12] Tim kasse, “Actin focused assessment for software process improvement,” Artech house, 2002.
[13] Paulish, and Carleton, “Case studies of software process improvement measurement,” Computer, Vol.27, No.9, 1994.

이 은 서
e-mail : eslee@andong.ac.kr
2001년~현 재 ISO/IEC 15504 국제 심사원
2004년 중앙대학교 컴퓨터공학과(박사)
2004년~현 재 임베디드 산업협회 전문위원
2004년~현 재 한국정보통신기술협회 위원
2005년~2007년 숭실대학교 정보미디어기술연구소 연구 교수
2008년 3월~현 재 안동대학교 컴퓨터공학과 조교수
관심분야 : CBD, Formal method, Quality model, SPI(Defect Analysis)

김 중 수

e-mail : kimjs@andong.ac.kr

1987년 안동대학교 전임강사

1996년 경북대학교 전자계산기(박사)

2010년~현 재 안동대학교 컴퓨터공학과 교수

관심분야 : 영상처리, 데이터베이스, 소프트웨어 엔지니어링