일정관리를 위한 Opportunity Tree 및 알고리즘 설계

이 은 서[†] · 이 상 호[††]

요 약

소프트웨어 개발 시, 일정과 품질에 저해 요인이 되는 결함이 다수 존재한다. 일정과 품질의 저해 요인을 제거하고 동시에 체계적으로 이를 관리하기 위하여 본 논문에서는 opportunity tree 프레임워크를 설계한다. 유사한 프로젝트를 수행 시 영역 전문가의 지식을 활용한 opportunity tree는 발생되는 문제점을 예측, 대비할 수 있게 하여, 소프트웨어 프로세스를 개선할 수 있다. 본 연구에서는 소프트웨어 개발 시 발생하는 일정관리에 대한 결함을 찾아내고, 원인을 식별 및 해결책을 제시하고자 한다.

키워드 : 소프트웨어 프로세스 개선, 프로젝트 일정관리, 품질, 프로젝트 관리, 소프트웨어 프로세스 OT, OT 프레임워크

Opportunity Tree and Algorithm design to schedule management

Lee Eun Se[†] · Lee Sang Ho[††]

ABSTRACT

There are many defects that cause the schedule and quality problems during software development. This paper designs the opportunity tree framework that removes and manages the schedule and quality problems as well. For the similar projects, we can estimate defects and prepare to solve them by using domain expert knowledge and the opportunity tree framework, which can greatly improve the software process. This research provides solution of schedule defect problem and detection of defect and its causes that happen on software development.

Key Words : SPI, Project Schedule Management, Quality, Project Management, Software Process Opportunity Tree, Opportunity Tree Framework

1. 연구 배경

소프트웨어 공학의 가장 중요한 목표는 주어진 일정과 프로젝트 비용 내에서 고품질의 시스템 및 제품을 생산해 내는 것이다. 이와 같은 목표를 달성하기 위해서 소프트웨어 개발자 및 프로젝트 관리자는 소프트웨어 프로세스의 전후 관계를 기반으로 하여 효율적인 개발 방법론과 관리 체계를 적용해야 한다. 또한 개발자 및 프로젝트 관리자는 영역 지식을 습득하여 시스템과 프로그램을 개발하여야 한다. 이와 같은 과정에서 영역 지식 습득이라는 것을 간과하는 경우가 많이 발생하고 있다. 이와 같은 문제는 프로그램 상에서 기능적인 문제와 프로그램 실행의 논리적인 문제로 발생을 하게 된다[1]. 따라서 영역 지식의 습득은 소프트웨어 개발에서 필수적인 요소가 되고 있으며 개발되는 소프트웨어의 품질과 직결되고 있다[6]. 그리고 훌륭한 소프트웨어 엔지니어는 분석과 설계 모형의 품질, 원시코드, 그리고 소프트웨어가 공학화되어 생성한 테스트 사례 등을 평가하는 측정을 한다. 이러한 실시간 품질평가를 달성하기 위해 엔지니어는 주관적인 방법보다 객관적인 방법으로 품질을 평가하는 테크니컬 측정(technical measures)을 사용해야 한다[1, 4, 6]. 품질을 측정하기 위해서는 많은 방법들이 존재한다. 그 중에서 발생된 결함(문제점)들을 제거하는 효율성(DRE : Defect Removal Efficiency) 검사 방법을 채택하여 설계된 opportynity tree(이하 OT)의 신뢰성을 측정하고자 한다.

신뢰성 있는 소프트웨어를 개발하기 위해서는 개발 영역의 내용을 정확히 알고 있으며 발생 가능한 문제점의 해결책을 제시할 수 있어야 한다. 따라서 소프트웨어와 개발과정에 존재하는 결함을 찾아내고 이를 제거하는 것이 중요한 요인이 된다. 이와 같은 요인은 품질로 귀결되게 되는데, 품질은 비용, 일정과 함께 프로젝트의 성공을 결정하는 주요 요소이다. 소프트웨어 품질 개념은 쉽게 정의할 수 있는 것이 아니다. 소프트웨어는 다양한 품질 관련 특성을 가지고 있는데 품질을 위한 국제 표준도 있다[2].

그러나 실제로 품질 관리는 종종 결함의 주변에서 맴돈다. 그러므로 인도된 결함 밀도, 즉 인도된 소프트웨어의 결함의 개수/단의 크기를 품질의 정의로 사용하는데, 이것은

※ 본 연구는 한국학술진흥재단의 지원에 의하여 수행되었음.(KRF 2004-005-D00172)
† 정 회 원 : 숭실대학교 정보미디어기술연구소 연구교수
†† 종신회원 : 숭실대학교 컴퓨터학부 교수
논문접수 : 2005년 9월 14일, 심사완료 : 2005년 11월 10일

사실 현재 업계에서 표준 정의가 되고 있다[3, 4]. 그러므로 결함의 의미를 소프트웨어 결함과 소프트웨어가 고객의 필요와 요구 사항과는 다르게 동작하게 하는 원인이라고 할 수 있다. 따라서 본 논문에서는 신뢰성 있는 소프트웨어를 생산하기 위하여 프로그램과 프로젝트를 구성하는 프로세스를 중심으로 하여 이를 개선하고자 소프트웨어 프로세스 개선 OT 프레임워크를 설계하고자 한다.

2. 기본 개념

2.1 결함

결함 정보는 다른 유형의 원시데이터이며, 소프트웨어 프로젝트에서 매우 중요한 것이다. 결함은 소프트웨어의 품질과 직접 관련이 있으므로 여러 의미에서 결함 데이터는 공수 데이터보다 더 중요하다[11, 12].

결함 데이터는 우선 프로젝트 관리를 위해 필요하다. 대규모 프로젝트는 수천개의 결함을 포함할 수 있는데, 이 결함은 다른 사람들이 프로젝트의 각각 다른 단계에서 발견하게 된다. 흔히 프로세스에서 결함을 고치는 사람과 결함을 발견하거나 보고하는 사람은 서로 다르다[13]. 보통 프로젝트는 소프트웨어가 최종 인도 전에 발견한 모든 또는 대부분의 결함을 제거하려고 할 것이다. 이런 시나리오에서 결함 보고와 해결은 비공식적으로 수행할 수 없다. 비공식적인 메커니즘의 사용은 발견한 결함에 대해 잊어버리는 결과를 가져올 수 있으므로 결함을 제거하지 않거나 다시 찾는데 추가 공수가 들어갈 수도 있다. 그러므로 적어도 결함을 기록해야 하고 해결할 때까지 추적해야 한다. 이런 절차를 위해서 결함의 징후, 의심되는 결함의 위치, 발견자, 해결자 등과 같은 정보가 필요할 것이다. 따라서 결함이란 프로젝트의 작업 산출물에서 발견되는 것으로 이 때문에 프로젝트의 목표를 달성하는데 부정적인 영향을 끼칠 수도 있다[14, 15].

결함에 대한 정보는 여러 가지 다양한 결함 검출 활동을 통해 발견한 결함의 개수도 포함 한다. 그래서 요구 사항 검토, 설계 검토, 코드 검토, 단위 테스트, 그리고 다른 단계에서 발견된 모든 결함 등을 기록한다. 프로젝트의 서로 다른 단계에서 발견된 결함 개수의 분포 데이터 역시 프로세스 역량 기준선(Process Capability Baseline) 생성에 사용한다[16]. 아울러 PDB(Process Data Base) 입력항목에 몇 가지 설명이 기록되는데, 예측에 대한 설명과 위험 관리에 대한 설명이 해당된다.

2.2 OT(Opportunity Tree)의 필요성

실제 프로젝트 수행 시에, 관리자들은 결함을 관리한다. 그리고 개발일정에 맞추기 위하여 많은 수의 결함들은 완전히 조치를 취하지 못하고 다음 개발 단계로 진행되기도 한다. 이와 같은 현상은 많은 프로젝트에서 발생하고 있다. 그 이유는 관리자의 경험부족과 일정상의 여유가 없기 때문인 경우가 대다수이다.

결함을 체계적으로 관리한다는 것은 결함을 분류하고 원인을 찾을 수 있으며, 이에 대한 대처 및 예방을 할 수 있는 것을 의미한다. 따라서 결함을 체계적으로 관리하는 것이 필요하게 되는데, 이러한 체계적인 관리를 위하여 OT (Opportunity Tree)를 사용하게 된다[28].

OT(Opportunity Tree)는 해결하려는 목표를 설정하게 되고, 전체 목표를 해결하기 위하여 요구되는 하위 목표를 결정하게 된다. 즉, 하위 목표를 달성하게 되면 상위 목표가 달성될 수 있게 된다. 그리고 목표 설정과 함께 목표를 해결하기 위하여 요구되는 문서와 전문가의 노하우를 OT(Opportunity Tree)화 하여 사용자에게 지침을 제공하고자 하는 것이 OT(Opportunity Tree)의 목적이 된다. 따라서 본 논문에서는 결함 관리를 전체 목표로 하여, 이를 달성하기 위한 하위 목표를 결정하고 해결책을 제시하여 사용자가 결함을 관리할 수 있도록 하는 것이 최종 목표가 된다. 또한 유사한 프로젝트를 시작하는 경우, OT(Opportunity Tree)를 활용하여 결함을 예방하고, 발생될 수 있는 결함을 예측하여, 이를 대처하기 위하여 일정 계획 수립 시, 참조가 될 수 있다.

2.3 소프트웨어 프로세스 개선의 필요성

프로세스를 개선하는 합리적인 방법은 프로세스의 특정한 속성들을 측정하고, 이들 속성에 근거해서 의미있는 메트릭스의 집합을 개발한 후에 특성에 관한 전략을 이끌어 내는 지표를 제공하기 위해서 이들 메트릭스를 사용하는 것이다. 그러나 우리가 소프트웨어 메트릭스와 소프트웨어 프로세스 개선에 있어서 영향을 말하기 전에, 프로세스는 소프트웨어 품질과 조직의 성능을 개선하는데 제어할 수 있는 많은 인자중의 하나라는 것이다[25, 26]. 따라서 소프트웨어 품질에 많은 영향을 주는 인자가 결함관리가 된다. 소프트웨어 품질과 조직의 성능에 영향을 주는 것으로는 프로세스를 기반으로 제품, 사람, 기술이 된다. 또한 제품, 사람, 기술이 프로세스를 일반화하는 과정에서 많은 결함들이 발생하여서 전체적인 소프트웨어 프로세스의 수준을 격하 시키게 된다. 그러므로 결함 관리의 필요성이 요구되게 된다.

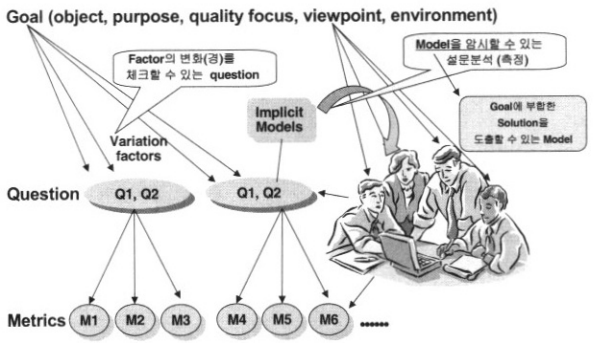
프로세스 개선을 위하여 소프트웨어의 실패 분석이 필요하며, 이를 위하여 다음과 같은 방법으로 수행된다.

- 모든 오류와 결함은 원인으로 분류한다.
- 각 오류와 결함을 수정한다.
- 각 범주에 속하는 오류와 결함을 계산하여 내림 차순으로 정리한다.
- 결과 자료는 조직에 가장 많은 비용이 드는 범주를 발견하기 위해서 분석한다.
- 계획은 가장 비용이 드는 오류와 결함의 종류를 제거할 의도로 프로세스를 수정하기 위해 개발된다.

본 논문에서는 위와 같은 소프트웨어 실패 분석 방법을 기반으로 일정 관리를 수행하기 위한 OT 프레임워크를 설계하고자 한다.

2.4 GQM(Goal / Question / Metrics)방법

본 논문에서는 자료의 수집과 분석을 GQM 방법에 의하여 수행하였다. GQM 접근 방법은 3단계로 정의되는데 구성은 (그림 1)과 같다.



(그림 1) GQM 방법

첫 번째 단계는 개념 적인 단계로서 단계의 요소로는 대상(Object), 목적(Purpose), 관점(View point), 초점(Focus)등이 되고 이 단계는 목표(Goal)를 정의하는 단계가 된다[22, 23].

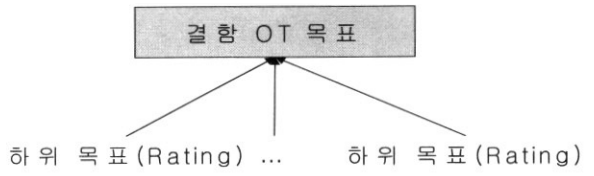
두 번째 단계는 운용단계로서 설정된 목표에 각각의 관점에서 정의 된 모델을 사용하여 합당한 질문(Question)을 하는 단계이다[23].

세 번째 단계는 정량적인 방법(Metrics)으로 질문에 답하는 단계를 말한다. 위의 3단계를 통해서 메트릭의 체계가 만들어 진다. 이러한 메트릭은 프로세스 개선을 위한 측정 도구로서 사용될 수도 있고 또한 경험 산출 요소(Experience Factory)의 기반이 되는 프로젝트 정보가 된다[21, 23].

2.5 계층적인 의사 결정 방법

본 논문에서는 식별된 OT의 목표 중에서 이를 달성하기 위한 항목을 결정하게 된다. 그러므로 목표 달성을 위하여 필요로 하는 하위 목표를 분석하여 계량화하기 위하여 의사결정을 위한 계층구조를 도식한다.

각 단계에서 항목이 차지하는 비중을 6개의 범위로 등급화 하였다. 등급화된 항목은 비교 메트릭스[31]를 기반으로 분석이 되며, 각 항목의 기하평균을 구하여 항목이 결함원인과의 연관성을 식별하게 된다. 결함 등급으로는 VL(Very Low), L(Low), N(Nominal), H(High), VH(Very High), EH(Extra High)의 6개 등급으로 세분화 하였다. 6개 등급 중에서 EH로 갈수록 우선순위가 높아지게 된다. 각 단계별 의사결정 구조도는 다음과 같다.



(그림 2) 원인 등급 구조도

(그림 2)에서는 찾고자 하는 결함 원인의 대상을 결함 원인 Rating에 기록을 한다. 그리고 각 결함 내용을 항목으로 나타내고 우선순위를 Rating에 6개 등급으로 서술하게 된다.

목표의 원인을 비교하기 위하여 메트릭스를 이용한다. 메트릭스는 각 중요도를 1부터 9까지 나열하여 관계를 규명하고자 한다. 비교 메트릭스의 측정 기준표는 <표 1>과 같다.

〈표 1〉 비교 메트릭스의 측정 기준표

(a_{ij})	정의
1	↑ 약 — 중요도가 같음
3	
5	중요도
7	
9	↓ 강 — 가장 중요도가 강함
2, 4, 6, 8	중간값
상호적인 관계	If $a_{ij}=w_i/w_j$, then $a_{ji}=1/a_{ij} = w_j/w_i$

<표 1>을 기준으로 각 단계별 항목간의 중요도를 계량화해서 나타낸다. 표에서 i와 j는 행렬을 나타내기 위한 인덱스이고 W_i와 W_j는 가중치를 적용한 후에 행렬을 나타낸 값이다.

OT 항목들은 <표 1>을 기준으로 분석하였다. 이때 각 단계의 항목등급의 차이를 비교하게 된다. 예를 들면, 한 항목이 EH이고 다른 항목이 N이면 두 항목 사이의 중요도는 두 등급 차이가 존재하기 때문에 중요도 차이는 3배가 발생하게 된다. 이와 같은 관계는 <표 2>과 같다.

〈표 2〉 등급 대비 중요도 차이 비율 산출표

항목간의 등급차이	등급 대비 중요도 차이비율
1	2배
2	3배
3	4배
4	5배
5	6배

<표 2>에 의하여 <표 1>의 비교 메트릭스를 산출할 수 있다. 그 내용은 <표 3>과 같다.

〈표 3〉 결함 관리 OT 항목간 비교 메트릭스의 예

	a11	a12	a13	a14	a15
a11	1	2	4	4	6
a12	2	1	2	2	4
a13	1/4	1/2	1	1	2
a14	1/4	1/2	1	1	2
a15	1/6	1/4	1/2	1/2	1

<표 3>에서는 a11, a12, a13, a14, a15 열을 기준으로 다른 항목과 중요도를 비교하게 된다. 예를 들면, a11과 a12는 a11이 결함항목으로서 2배가 많게 분석이 되었다. 그 이유

는 a11은 EH이고, a12는 VH이므로, <표 2>에 의하여 항목 간의 등급이 1차이가 발생하였다. 따라서 그에 따른 등급 대비 중요도 차이 비율은 2배가 된다. 결론적으로 a11은 a12 보다 결함을 발생시킬 확률이 2배 많다는 의미가 된다.

<표 1>에서 산출된 비교 메트릭스를 기반으로 각 단계별 항목의 중요도를 산출하고자 한다. 각 항목의 중요도가 높을수록 결함의 원인이 될 확률이 높으며, 산출은 기하 평균을 이용하여 산출하였다.

각 단계별 산출표는 <표 4>와 같다.

〈표 4〉 결함 원인 중요도 산출의 예

항목	산출값
a11	(1x2x4x4x6)1/5 = 1921/5 = 2.862
a12	(2x1x2x2x4)1/5 = 321/5 = 2
a13	(1/4 x1/2x1x1x2)1/5 = 0.251/5 = 0.758
a14	(1/4x1/2x1x1x2)1/5 = 0.251/5 = 0.758
a15	(1/6x1/4x1/2x1/2x1)1/5 = 0.101/5 = 0.631

각 단계의 항목을 기하 평균에 의하여 산출하였다. 산출 결과를 분석하면 요구사항 분석 단계의 경우, a11과 a12 항목이 결함 원인이 될 수 있다고 분석되었다. 이와 같은 분석 자료를 이용하여 결함을 해결하기 위한 내용의 우선순위를 결정할 수 있다.

3. 본론 및 사례연구

본 장에서는 회사에서 실제 프로젝트를 수행하는 과정에서 발생하는 소프트웨어 프로세스의 문제점을 해결하기 위한 개선 OT를 제공하고자 한다. 또한 일정 관리를 위한 OT 설계 알고리즘을 제안하고자 한다. 이를 위하여 프로세스 개선을 위한 항목을 찾아야 한다. 따라서 프로세스 개선에서 요구되는 항목을 찾기 위하여 설문서를 작성하였다. 또한 문제점을 검출하기 위한 설문서의 구조와 내용을 설명하고, 결과를 계층적 의사 결정법에 의하여 분석하게 된다. 그리고 분석된 결과를 결함 처리 방안을 시스템화하기 위하여 일정 관리를 위한 OT 프레임워크를 설계한다.

설문서 대상 회사 및 프로젝트의 도메인으로는 디지털 시스템을 대상으로 수행하였으며, 네 개회사의 21개 프로젝트를 대상으로 하였다.

3.1 소프트웨어 프로세스 개선을 위한 OT 프레임워크의 개념 및 필요성

소프트웨어 프로세스 개선을 위해서 요구되는 항목으로는 일정, 비용 및 품질의 척도가 있다. 본 논문에서는 세 가지 척도 중에서 일정과 품질을 대상으로 하였다. 비용에 관해서는 향후 연구 내용으로 설정하였다. 따라서 일정 척도에 대한 내용은 설문조사에 의하여 결정되었으며, 21개의 프로젝트 응답자가 세 가지 요인의 필요성을 제시하였다.

〈표 5〉 결함 처리를 위한 요구사항 항목 분석표

전체 목표	결함 관리
하위 목표	결함 자료 수집 및 식별
	결함 연관성 분석
	잔존 결함 수정
	결함 제거
	결함 예방

품질과 연관된 결함 관리 OT는 <표 5>와 같은 내용을 설계하였다[28].

결함 관리 OT는 다음의[28] 연구에서 제시가 되었고 본 논문에서는 일정과 품질에 초점을 맞추어서 제시를 하였다.

OT 프레임워크는 기존의 시스템에서 필요성을 확인할 수 있다. 그 필요성은 다음과 같이 제시할 수 있다.

특정 도메인의 정보를 이용하기 위해서는 도메인 전문가의 도움을 받아서 수동적으로 정보를 찾게 된다. 그리고 도메인 전문가에게 정보를 찾기 위하여 도움을 요청하게 되고, 요청에 대한 내용은 게시판을 이용하여 질문을 하게 된다. 따라서 그 결과를 응답받기 위해서는 많은 시간을 기다려야 한다. 또한 도메인 전문가도 요청되는 정보를 찾기 위하여 수동적인 절차에 의하여 정보를 찾아서 제공하게 된다. 따라서 수동적인 절차를 자동화 및 웹에서 제공을 하고 필요한 정보를 검색할 수 있게 한다. 그러므로 검색을 위하여 사용자가 사용하기 쉬운 인터페이스를 제공해야 하며, 재사용성을 위하여 활용하고자 하는 분야의 자료를 통합 및 관리하는 방법이 제공되어야 한다. 또한 소프트웨어 프로세스 개선을 하기 위해서는 개선 항목을 지정하여 지속적으로 항목의 변경 및 추가 사항이 존재하는가를 검사해야 한다. 이를 통하여 정확하고 신속한 소프트웨어 프로세스 개선을 수행할 수 있게 된다.

일정관리를 수행하지 않았을 경우, 다음과 같은 문제점을 발생시킬 수 있다.

- 프로젝트 수행 시, 잠재되어 있는 문제점에 의한 일정 지연 발생 및 예측과 대비 미흡, 추가비용 발생, 전체적인 품질 저하
- 개선되지 않은 프로세스에 의하여 개발되는 소프트웨어에 문제점이 항상 내재됨
- 결함 발생 시, 이를 해결하기 위한 방법 부재 및 원인 파악의 난해성
- 유사한 프로젝트 수행 시, 동일한 내용의 결함 발생

따라서 문제점들을 완화시키기 위하여 일정 관리를 위한 OT를 제안하고자 한다.

3.2 일정 관리를 위한 OT 알고리즘

일정 관리를 위한 OT 알고리즘은 GQM방식을 기반으로 하여 다음과 같이 제시하였다.

- 찾고자 하는 목표 선정
- 목표를 달성하기 위한 하위 목표 선정
- 목표와 하위 목표의 타당성 및 신뢰성 검증을 위한 설문서 작성
- 목표와 하위 목표를 결정하기 위한 의사 결정(본 논문에서는 계층적 의사 결정법을 사용하였음)
- 목표와 하위 목표 선정후 해결책 제시
- 선정된 항목을 OT로 구현

본 논문에서는 위에서 제시한 알고리즘을 기반으로 하여 일정 관리를 위한 OT를 설계하였다. 설계 내용은 다음과 같다.

3.3 시스템 환경 분석

3.3.1 시스템 구성인자 식별

일정 관리를 위한 OT를 설계하기 위해서는 선행 작업으로서 시스템 환경의 분석이 요구된다. 시스템 환경은 일정 관리를 위한 OT가 구동될 하드웨어 기반의 환경을 분석하는데 목적이 있다. 따라서 시스템 환경을 파악하기 위하여 시스템 구성인자를 식별하게 되었다.

시스템 환경 설정 단계에서는 시스템 구성인자간의 연관성을 추출하여 전체적인 시스템의 구성인자를 식별하게 된다. 또한 시스템의 연관성 추출을 기반으로 기능적인 요구사항의 충돌 및 실현가능성을 판별할 수 있다. 그러므로 시스템 구조도에서는 시스템 요소간의 관계를 명시함으로서, 연관성을 쉽게 인식할 수 있다.

본 논문에서는 일정 관리를 위한 OT를 설계하기 위한 관점으로 시스템 환경을 분석하였다. 시스템 구성인자는 일정 관리를 위한 OT를 구축하기 위하여 요구되는 하드웨어를 식별하는 것이다. 따라서 식별된 하드웨어 환경 하에 일정 관리를 위한 OT가 구축될 수 있게 된다. 시스템 구성인자로는 보안을 처리할 서버와 사용자가 일정 관리를 위한 OT를 접근하여 사용할 수 있게 해주는 웹 서버, 그리고 일정 관리를 위한 OT에서 자료를 참조하는 데이터 베이스를 구성인자로 추출하였다. 식별된 시스템 구성인자는 (그림 3)과 같다.

3.3.2 스테이크 홀더 정의

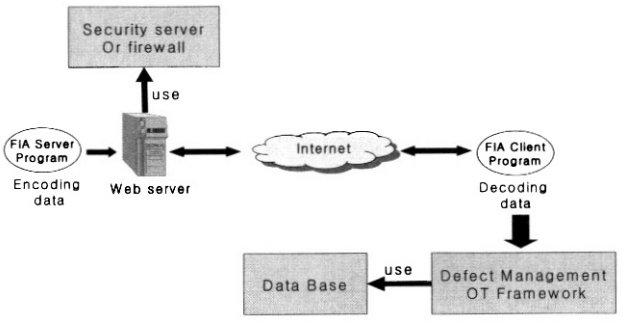
시스템의 환경을 식별하여 구성인자를 파악하는 과정에서 추가되어야 할 사항으로는 스테이크 홀더를 식별하는 것이다. 스테이크 홀더는 시스템과 이후의 요구사항 추출 및 분석에서 중요한 역할을 하게 된다. 추출된 비 기능적인 시스템 요소와 기능적인 요구사항은 스테이크 홀더를 통하여 연결되게 된다. 그러므로 스테이크 홀더의 식별 및 정의는 시스템 요소와 요구사항 분석에 많은 영향을 주게 된다. 본 논문에서는 일정 관리를 위한 OT를 설계하는 것이 목적이며, 이를 위하여 다음과 같은 스테이크 홀더를 추출할 수 있었다. 추출된 스테이크 홀더의 정의는 <표 6>과 같다.

〈표 6〉 스테이크 홀더의 정의

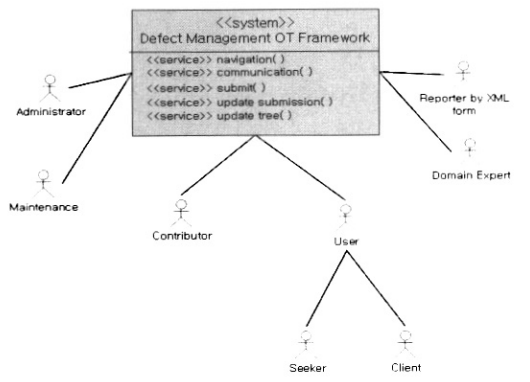
스테이크 홀더 명	설 명
설문서 작성자	OT를 개발하기 위하여 설문서를 작성하여 초기 요구사항을 조사하는 사람
개발자	OT를 개발하는 사람
프로젝트 관리자	소프트웨어 개발 및 프로젝트를 관리하며 문제점 발생시 이에 대한 해결책을 결정하는 사람
프로세스 개선 담당자 (형상관리 및 품질 보증 담당자)	OT 내용이 클라이언트에게 전달되기 이전에 발생되는 문제를 담당하는 사람, (예-결함관리 OT 개발과 연관하여 요구사항이 올바르게 구현되는가를 검사하고, 잘못된 부분을 찾아내는 사람(독립된 조직), 본 개발에서는 리뷰방법을 이용함)
클라이언트	OT 개발을 요구하는 사람
사용자	원하는 정보를 얻기 위하여 탐색기능을 이용하는 사람
정보 추가자	OT에 추가적으로 발생하는 정보를 추가 시키는 사람, (연구단체)
관리자	OT의 계층을 정렬하고, 시스템에 접근 가능하도록 관리하는 사람
도메인 전문가	OT와 연관된 지식을 가진 사람
소프트웨어 유지보수 담당자	OT 내용이 클라이언트에게 전달되면서 발생되는 문제를 담당하는 사람

추출된 스테이크 홀더는 도메인 지식을 시스템화하려는 관점의 기능적 요구사항과 시스템의 비 기능적 요구사항에 연관된 인력자원을 주목적으로 추출하였다.

또한 각 스테이크 홀더와 시스템의 연관관계에서 일정 관리를 위한 OT의 시나리오를 추출할 수 있다. 추출된 시나리오는 요구사항을 기능화하기 위하여 필수적으로 요구되는 것으로서, 기능들 간의 실현 가능 타당성을 재시할 수 있게 된다. 각 스테이크 홀더의 연관관계는 (그림 4)와 같다.



(그림 3) 시스템 구조도



(그림 4) 스테이크 홀더 구조도

3.4 요구사항 추출

요구사항 분석을 위한 설문서는 GQM 방식에 의하여 작성되었으며, 이를 기반으로 사용자가 요구하는 기능을 파악하고자 하는 목적이 있다. 우선 목표(Goal)는 결함 관리라는 것을 수행하기 위하여 필요로 하는 하위 목표가 무엇인가를 결정해야만 한다. 따라서 하위 목표는 전체 목표를 달성하기 위한 단위로 간주 할 수 있다.

처음 설문서는 전체 목표을 완성하기 위한 하위 목표를 식별하는데 목적이 있다.

일정 관리를 위한 OT 요구사항 추출 설문서는 <표 7>과 같다.

〈표 7〉 일정 관리를 위한 OT 요구사항 추출 설문서

소프트웨어 프로세스 개선을 위한OT 프레임워크 요구사항 추출 설문서	
일반적인 사항	1. 작성 일자 :
	2. 작성자 직급 :
	3. 프로젝트명 :
	4. 프로젝트 특성 분류:
프로세스 개선의 이슈사항	1. 이전과 현재의 프로젝트 수행 경험에 기반하여 프로세스중에서 중요한 이슈(Issue)가 되는 항목을 순서대로 나열하시오.
	2. 1번 항목중에서 여러 프로젝트에 공통적으로 발생한 항목은 무엇입니까?

결함 처리의 사항 1번 설문내용은 결함처리를 하기 위하여 요구되어지는 사항들이 무엇인가를 조사하기 위함이다. 그 결과 다음과 같은 통계 분석 자료를 추출하였다. 전체 응답은 152명을 대상으로 수행하였으며, 오류 응답은 존재하지 않았다. 빈도를 상세화 하면 <표 8>과 같다.

〈표 8〉 일정 관리를 위한 OT 항목 빈도 분포 표

	Frequency	Percent
일정관리를 위한 생명주기 선택	50	33
프로젝트 계획 및 위험 요소 식별	56	36.9
일정 지연을 예방하기 위한 요소들의 타당성 검증	36	23.5
일정관리를 위한 기구 마련	10	6.6
Total	152	100.0

분석된 일정 관리를 위한 OT 항목 빈도에 의하여 여러 항목 중에서 "일정 관리를 위한 기구 마련"을 제외한 나머지 항목에 대하여 필요성이 있다고 응답을 했다. 따라서 "일정 관리를 위한 기구 마련"을 제외한 이외의 항목이 일정 관리를 위한 OT의 항목이 대상이 되었다. 그러므로 이후의 일정 관리를 위한 OT의 목표는 분석된 항목을 대상으로 수행하게 된다. <표 8>에서 일정 관리를 위한 전체 목표를 결정하였다. 그리고 각각의 목표를 실현하기 위하여 어떤 문제점들이 발생하는지 파악할 필요성이 있다. 또한 식별된 문제점들은 일정 관리를 위한 OT를 위한 목표 중에서 어느 범주에 속하는지 구별을 하였다.

각각의 결함 처리 OT 항목 내용은 <표 9>와 같이 설명된다.

〈표 9〉 일정 관리를 위한 OT 항목의 내용

일정관리를 위한 OT	내 용
일정관리를 위한 생명주기 선택	일정관리를 수행하기 위한 생명주기를 정의함
프로젝트 계획 및 위험 요소 식별	프로젝트의 수행 계획을 수립하고 위험 요소를 파악하여 대비함
일정 지연을 예방하기 위한 요소들의 타당성 검증	일정 지연 요소를 파악하여 이를 예방하기 위한 타당성을 검증함
일정관리를 위한 기구 마련	일정관리를 수행하기 위하여 전사적인 조직체계를 마련함

우선 프로젝트 수행 시 발생되는 문제점들은 <표 10>에 제시하였으며, 빈도분석도 병행하여 수행하였다. 분석 결과는 여러 문제점들 중에서 프로젝트의 규모 및 복잡도에 따른 생명주기 모델 설정 필요성이 10.5%를 차지하였고, 기술적 타당성 검증(사례연구, 실패사례 연구, 모의 실험, 프로토타이핑) 과 시스템의 필요성, 목표, 제약사항 및 요구되는 기술정의가 10.1%로 많은 비중을 차지하고 있었다.

<표 10>에서 제시된 문제점들은 <표 9>에서 어느 범주에 속하는지 찾아서 연결을 하였다. 이와 같은 작업은 각각의 일정 관리 항목을 완성하기 위한 중요한 요소가 된다. 따라서 일정 관리 OT 항목과 세부 요소들의 연관성은 <표 11>과 같이 제시하였다.

〈표 10〉 프로젝트 수행 시 발생되는 문제점들

빈도 / 결함내용	Frequency	Percent
프로젝트에 맞는 테일러링 된 생명주기 생성	22	9.2
진척도 측정 방법(메트릭, 기술적 및 관리적 검토, 수립된 품질 및 수행 기준의 심사)	18	7.6
개발환경 구축	15	6.3
기술적 타당성 검증(사례연구, 실패사례 연구, 모의 실험, 프로토 타이핑)	24	10.1
제품의 인도시기 설정	12	5.0
시스템의 필요성, 목표, 제약사항 및 요구되는 기술정의	24	10.1
프로세스 및 생성되는 산출물들 간의 연관성 식별	22	9.2
경제적 타당성(투자 효율성, 시장성, 비용과 수익의 비교)	21	8.8
프로젝트의 규모 및 복잡도에 따른 생명주기 모델 설정	25	10.5
업무 분해도 및 산정치, 기반구조 요소를 근거로 하여 프로젝트 일정 수립	17	7.1
법적 타당성(사용도구들의 법적 권한, 시장 및 관행들에 대한 조사)	15	6.3
프로젝트 계획이 실무자에게 인식되어 활용할 수 잇는 체계 마련	11	4.6
수립된 품질 기준 심사	12	5.0
Total	188	100.0

〈표 11〉 일정 관리 항목에 따른 중요 항목 내용 분류표

결함 처리 OT 항목	결함 내용
일정관리를 위한 생명주기 선택	1. 프로젝트에 맞는 테일러링 된 생명주기 생성 2. 진척도 측정 방법(메트릭, 기술적 및 관리적 검토, 수립된 품질 및 수행 기준의 심사) 3. 프로젝트의 규모 및 복잡도에 따른 생명주기 모델 설정
프로젝트 계획 및 위험 요소 식별	1. 개발환경 구축 2. 시스템의 필요성, 목표, 제약사항 및 요구되는 기술정의 3. 프로세스 및 생성되는 산출물들 간의 연관성 식별 4. 업무 분해도 및 산정치, 기반구조 요소를 근거로 하여 프로젝트 일정 수립 5. 프로젝트 계획이 실무자에게 인식되어 활용할 수 잇는 체계 마련
일정 지연을 예방하기 위한 요소들의 타당성 검증	1. 기술적 타당성 검증(사례연구, 실패사례 연구, 모의 실험, 프로토 타이핑) 2. 경제적 타당성 검증(투자 효율성, 시장성, 비용과 수익의 비교) 3. 법적 타당성 검증(사용도구들의 법적 권한, 시장 및 관행들에 대한 조사) 4. 수립된 품질 기준 심사
일정관리를 위한 생명주기 선택	1. 제품의 인도시기 설정

<표 11>의 내용은 이후의 개발 단계에서 일정 관리를 위한 OT 항목 구현 시 완성되어야 할 핵심 내용이 된다. 분석된 하위 목표는 <표 12>와 같다.

〈표 12〉 일정 관리 OT 설계를 위한 요구사항 항목 분석표

전체 목표	일정 관리
하위 목표	일정관리를 위한 생명주기 선택
	프로젝트 계획 및 위험 요소 식별
	일정 지연을 예방하기 위한 요소들의 타당성 검증
	일정관리를 위한 기구 마련

3.5 일정 관리를 위한 OT 설계

일정 관리를 위한 OT 프레임워크에서 각 OT 항목을 가시화하고 해당 OT 항목을 해결하기 위한 활동과 방향을 제시하고자 한다. 따라서 OT 항목은 구조화된 형태로 제시되어야 전체 구조의 식별과 이해가 가능할 수 있다. 본 논문에서는 일정 관리를 위한 OT 프레임워크의 구성을 트리형태로 설계하였다. 또한 각 OT 항목을 상세히 설명하고자 <표 13>과 같이 구성하였다.

〈표 13〉 일정 관리를 위한 OT 프레임워크 항목

항목 이름	내용
Parent	상위 목표를 나타냄
Children	하위 목표를 나타냄
Activity	해당 OT 항목과 연관된 생명주기 활동을 나타냄
Solution	해당 OT 항목을 성취하기 위한 방법 및 활동을 나타냄
Target Benefit of the Solution	해당 OT 항목 사용의 효과를 사용자에게 알려주기 위하여 Quality, Delivery, Cycle time, Waste로 구분하여 표시함
Precondition	해당 OT 항목을 수행하기 위한 선행 조건을 나타냄
Postcondition	해당 OT 항목을 완료한 후의 결과를 나타냄
Contract	해당 OT 항목을 수행하기 위한 제약 사항을 나타냄
Document	해당 OT 항목과 연관된 문서를 나타냄

3.5.1 일정 관리를 위한 생명주기 선택

일정관리를 위한 생명주기 선택의 요소 수집 및 식별에서 발생하는 내용은 다음과 같다. 등급은 빈도에 의한 내용으로 <표 14>와 같이 제시하였다.

〈표 14〉 일정관리를 위한 생명주기 선택의 항목의 내용

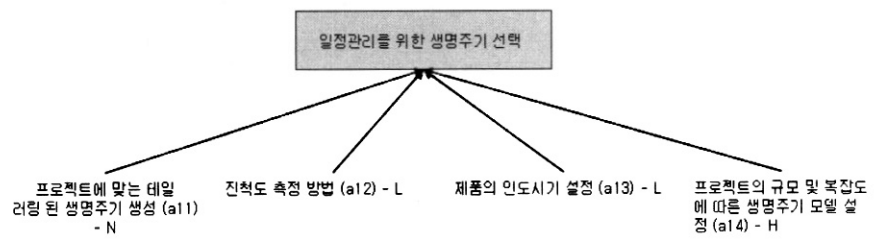
내용	중요도
1. 프로젝트에 맞는 테일러링 된 생명주기 생성	Nominal
2. 진척도 측정 방법(메트릭, 기술적 및 관리적 검토, 수립된 품질 및 수행 기준의 심사)	Low
3. 제품의 인도시기 설정	Low
4. 프로젝트의 규모 및 복잡도에 따른 생명주기 모델 설정	High

결함 자료 수집 및 식별에 관한 의사결정 구조도와 등급화된 항목은 (그림 5)와 같다.

항목간 비교 메트릭스는 <표 15>와 같다.

〈표 15〉 일정관리를 위한 생명주기 선택 항목간 비교 메트릭스

	a11	a12	a13	a14
a11	1	2	2	1/2
a12	1/2	1	1	1/3
a13	1/2	1	1	1/3
a14	2	3	3	1



(그림 5) 일정관리를 위한 생명주기 선택의 원인 등급 구조도

일정관리를 위한 생명주기 선택 항목간 비교 메트릭스 a11, a12, a13, a14 열을 기준으로 다른 항목과 중요도를 비교하게 된다. 따라서 a11과 a12는 a11이 2배 많게 분석이 되었다. a11은 N 이고, a12는 L이므로 두 항목간의 등급차이가 1이므로 중요도 차이 비율은 2배가 된다.

일정관리를 위한 생명주기 선택을 수행하기 위한 목표 중 어떤 목표가 중요한가를 산출하여야 한다. 따라서 이와 같은 중요한 목표를 산출하기 위하여 기하 평균을 이용하였다.

일정관리를 위한 생명주기 선택 항목의 목표 중요도는 <표 16>과 같이 산출되었다.

〈표 16〉 일정관리를 위한 생명주기 선택의 하위 목표 중요도 산출표

항목	산출값
a11	$(1 \times 2 \times 2 \times 1/2)^{1/4} = (1/3)^{1/4} = 1.19$
a12	$(1/2 \times 1 \times 1 \times 1/3)^{1/4} = 3^{1/4} = 0.64$
a13	$(1/2 \times 1 \times 1 \times 1/3)^{1/4} = 3^{1/4} = 0.64$
a14	$(2 \times 3 \times 3 \times 1)^{1/4} = (1/3)^{1/4} = 1.68$

a14(프로젝트의 규모 및 복잡도에 따른 생명주기 모델 설정) 하위 목표가 일정관리를 위한 생명주기 선택에서 가장 중요한 달성 목표로 분석되었다. 따라서 a14가 일정관리를 위한 생명주기 선택 목표를 달성하기 위하여 우선순위가 가장 높고, 달성되어야 할 하위목표로 나타났다. 그리고 나머지 a12(진척도 측정 방법), a13(제품의 인도시기 설정)은 같은 중요도를 나타냈으며, a11, a12, a13 및 a14의 목표를 달성하기 위해서는 <표 17>과 같은 해결책을 제시할 수 있다.

〈표 17〉 일정관리를 위한 생명주기 선택 목표를 위한 해결책

내용	해결책
1. 프로젝트에 맞는 테일러링 된 생명주기 생성	1. 프로젝트 목적 식별 2. 생명주기별 특성 파악 3. 요구사항 분석
2. 진척도 측정 방법(메트릭, 기술적 및 관리적 검토, 수립된 품질 및 수행 기준의 심사)	1. 업무의 특성 분석 2. 기술 분석서 작성
3. 제품의 인도시기 설정	1. 개발단계마다 마일스톤 정의 2. 인도 시험을 위한 체크 리스트 정의 3. 인도 기준 정의
4. 프로젝트의 규모 및 복잡도에 따른 생명주기 모델 설정	1. 프로젝트별 세부 작업의 내용 파악 2. 작업간의 연관성 파악 3. 작업 및 기능의 충돌성 판별

3.5.2 프로젝트 계획 및 위험 요소 식별

프로젝트 계획 및 위험 요소 식별에서 발생하는 내용과 해결책은 다음과 같다. 등급은 빈도에 관한 내용으로 <표 18>과 같이 제시하였다.

〈표 18〉 프로젝트 계획 및 위험 요소 식별 항목의 내용

내용	중요도
1. 개발환경 구축	Low
2. 시스템의 필요성, 목표, 제약사항 및 요구되는 기술 정의	Very High
3. 프로세스 및 생성되는 산출물들 간의 연관성 식별	High
4. 업무 분해도 및 산정치, 기반구조 요소를 근거로 하여 프로젝트 일정 수립	Nominal
5. 프로젝트 계획이 실무자에게 인식되어 활용할 수 있는 체계 마련	High

프로젝트 계획 및 위험 요소 식별에 관한 의사결정 구조도와 등급화된 항목은 (그림 6)과 같다.

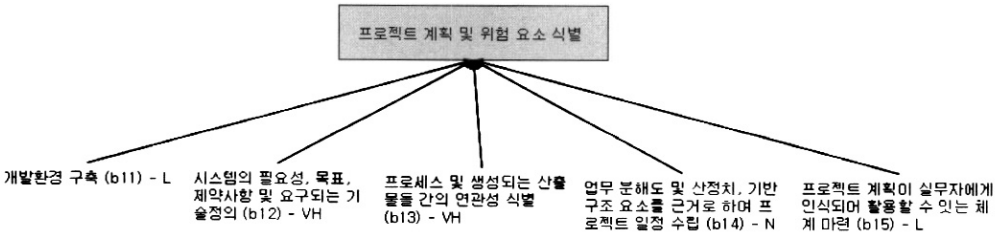
항목간 비교 메트릭스는 <표 19>와 같다.

〈표 19〉 프로젝트 계획 및 위험 요소 식별 항목간 비교 메트릭스

	b11	b12	b13	b14	b15
b11	1	1/4	1/3	1/3	1/3
b12	4	1	2	2	2
b13	3	1/3	1	1	1
b14	3	1/3	1	1	1
b15	3	1/3	1	1	1

프로젝트 계획 및 위험 요소 식별 항목간 비교 메트릭스는 b11, b12, b13, b14, b15 열을 기준으로 다른 항목과 중요도를 비교하게 된다. 따라서 b11과 b12는 b12가 4배 많게 분석이 되었다. 그 이유는 b11은 L 이고, b12는 VH이므로 두 항목간의 등급차이가 3이므로 중요도 차이 비율은 4배가 된다.

프로젝트 계획 및 위험 요소 식별을 수행하기 위한 목표 중 어떤 목표가 중요한가를 산출하여야 한다. 따라서 이와 같은 중요한 목표를 산출하기 위하여 기하 평균을 이용하였다. 프로젝트 계획 및 위험 요소 식별 항목의 목표 중요도는 <표 20>과 같이 산출되었다.



(그림 6) 프로젝트 계획 및 위험 요소 식별 원인 등급 구조도

〈표 20〉 프로젝트 계획 및 위험 요소 식별의 하위 목표 중요도 산출표

항목	산출값
b11	$(1 \times 1/4 \times 1/3 \times 1/3 \times 1/3)^{1/5} = (1/108)^{5}$ 0.39
b12	$(4 \times 1 \times 2 \times 2 \times 2)^{1/3} = 32^{1/5} = 2$
b13	$(3 \times 1/3 \times 1 \times 1 \times 1)^{1/5} = (1)^{1/5} = 1$
b14	$(3 \times 1/3 \times 1 \times 1 \times 1)^{1/5} = (1)^{1/5} = 1$
b15	$(3 \times 1/3 \times 1 \times 1 \times 1)^{1/5} = (1)^{1/5} = 1$

b12(시스템의 필요성, 목표, 제약사항 및 요구되는 기술정의) 하위 목표가 프로젝트 계획 및 위험 요소 식별에서 가장 중요한 달성 목표로 분석되었다. 따라서 b12(시스템의 필요성, 목표, 제약사항 및 요구되는 기술정의)가 프로젝트 계획 및 위험 요소 식별을 달성하기 위하여 우선순위가 가장 높고, 달성되어야 할 하위목표로 나타났다. 그리고 b11, b12, b13, b14, b15의 목표를 달성하기 위해서는 <표 21>과 같은 해결책을 제시할 수 있다.

〈표 21〉 프로젝트 계획 및 위험 요소 식별 목표를 위한 해결책

결함 내용	해결책
1. 개발환경 구축	1. 기능적 요구사항과 비기능적 요구사항 분서 2. 개발에 적절한 인력 마련 3. 프로젝트 수행에 필요한 인프라 구축
2. 시스템의 필요성, 목표, 제약사항 및 요구되는 기술정의	1. 프로젝트의 목표 분석 2. 프로젝트의 위험 요소 분석
3. 프로세스 및 생성되는 산출물들 간의 연관성 식별	1. 프로세스별 산출물 파악 2. 업무의 선후관계 분석 3. 작업의 우선순위 정의
4. 업무 분해도 및 산정치, 기반구조 요소를 근거로 하여 프로젝트 일정 수립	1. 프로젝트 일정 산정의 근거 정의 2. 프로젝트 일정 산정의 인프라 구축
5. 프로젝트 계획이 실무자에게 인식되어 활용할 수 잇는 체계 마련	1. 프로젝트 계획의 주기적인 확인 2. 일정단위별 진척도 측정

3.5.3 일정 지연을 예방하기 위한 요소들의 타당성 검증

일정 지연을 예방하기 위한 요소들의 타당성 검증에서 발생하는 내용과 해결책은 다음과 같다. 등급은 빈도에 관한 내용으로 <표 22>과 같이 제시하였다.

〈표 22〉 일정 지연을 예방하기 위한 요소들의 타당성 검증 항목의 내용

내용	중요도
1. 기술적 타당성 검증	Very High
2. 경제적 타당성 검증	Nominal
3. 법적 타당성 검증	Low
4. 수립된 품질 기준 심사	Low

일정 지연을 예방하기 위한 요소들의 타당성 검증에 관한 의사결정 구조도와 등급화된 항목은 (그림 7)과 같다.

항목간 비교 메트릭스는 <표 23>과 같다.

〈표 23〉 일정 지연을 예방하기 위한 요소들의 타당성 검증 항목간 비교 메트릭스

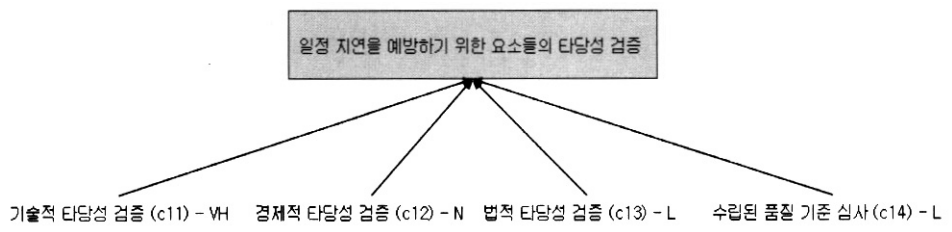
	c11	c12	c13	c14
c11	1	3	4	4
c12	1/3	1	2	2
c13	1/4	1/2	1	1
c14	1/4	1/2	1	1

일정 지연을 예방하기 위한 요소들의 타당성 검증 항목간 비교 메트릭스는 c11, c12, c13, c14 열을 기준으로 다른 항목과 중요도를 비교하게 된다. 따라서 c11과 c12는 c11가 3배 많게 분석이 되었다. 그 이유는 c11은 VH 이고, c12는 N이므로 두 항목간의 등급차이가 3이므로 중요도 차이 비율은 4배가 된다.

일정 지연을 예방하기 위한 요소들의 타당성 검증을 수행하기 위한 목표 중 어떤 목표가 중요한가를 산출하여야 한다. 따라서 이와 같은 중요한 목표를 산출하기 위하여 기하 평균을 이용하였다. 일정 지연을 예방하기 위한 요소들의 타당성 검증 항목의 목표 중요도는 <표 24>와 같이 산출되었다.

〈표 24〉 일정 지연을 예방하기 위한 요소들의 타당성 검증의 하위 목표 중요도 산출표

항목	산출값
c11	$(1 \times 3 \times 4 \times 4)^{1/4} = (48)^{1/4} = 2.63$
c12	$(1/3 \times 1 \times 2 \times 2)^{1/4} = (32)^{1/4} = 1.07$
c13	$(1/4 \times 1/2 \times 1 \times 1)^{1/4} = (1)^{1/5} = 0.59$
c14	$(1/4 \times 1/2 \times 1 \times 1)^{1/4} = (1)^{1/5} = 0.59$



(그림 7) 일정 지연을 예방하기 위한 요소들의 타당성 검증 원인 등급 구조도

c11(기술적 타당성 검증) 하위 목표가 일정 지연을 예방하기 위한 요소들의 타당성 검증에서 가장 중요한 달성 목표로 분석되었다. 따라서 c11이 일정 지연을 예방하기 위한 요소들의 타당성 검증을 달성하기 위하여 우선순위가 가장 높고, 달성되어야 할 하위목표로 나타났다. 그리고 c11, c12, c13, c14의 목표를 달성하기 위해서는 <표 25>과 같은 해결책을 제시할 수 있다.

〈표 25〉 일정 지연을 예방하기 위한 요소들의 타당성 검증 목표를 위한 해결책

내용	해결책
1. 기술적 타당성 검증	1. 유사한 프로젝트의 사례 분석 2. 실패한 프로젝트의 사례 분석 3. 기능의 구현 가능성 분석
2. 경제적 타당성 검증	1. 투자대비 효과에 대한 분석 방법 마련 2. 시장 조사에 의한 타당성 분석
3. 법적 타당성 검증	1. 요구되는 프로그램 및 지적 재산권의 라이센스 구입
4. 수립된 품질 기준 심사	1. 작업의 마일스톤마다 통과 기준 마련 2. 전사적인 기준에 의한 품질 측정 수행

4. 사례 연구

4.1 시스템 설계

OT를 프레임워크로 설계하기 위하여 개발환경을 정의하였다. 개발 환경에서 WAS의 기본이라고 할 수 있는 Tomcat을 사용하였다. 이는 상업용 WAS에서도 서브렛의 형태로 작동이 가능하기 때문이다. 단, 자바 기반이 WAS만 가능하다는 전제 조건이 있다. 그러나 자바는 속도면에서 느리지만 여러 플랫폼에서 사용이 가능하기 때문에 플랫폼 독립적이면 호환성이 높아지는 이점을 얻을 수 있게 된다. 또한 현재 WAS와의 연동이 안되는 것은 호환성면에서 많이 떨어지게 되어서 사용하지 못하는 경우가 많이 발생된다. 따라서 호환성을 고려하여 WAS 환경 하에서 개발하게 되었다.

데이터베이스 관리 시스템에서는 일반적으로 제공되는 함수를 사용하지 않고 기본적인 SQL문만을 사용하였다. 이와 같은 이유는 다른 데이터베이스 관리 시스템과도 테이블 이름등의 기본적인 정보만 같으면 연동이 가능해지기 때문이다. 또한 데이터베이스의 트랜잭션의 구현을 하나의 자바 서버 페이지로 처리했다. 하나의 서버 페이지로 구현을 하면, 차후 개발 및 유지 보수가 쉬워지게 되는 이점이 있다. 향후 트랜잭션이 복잡해지면 기능에 의한 분류가 요구되지만, 본 논문에서는 기본적인 트랜잭션을 사용하므로 하나의 서버 페이지로 처리하였다.

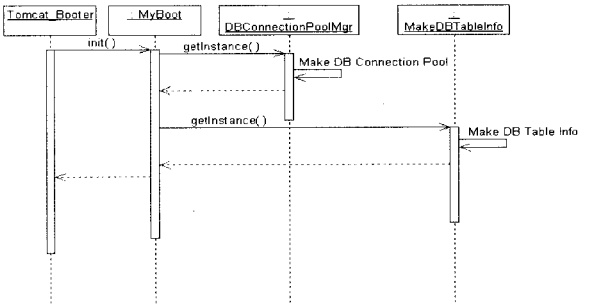
XML은 결과의 내용을 보고하는 형태로 사용된다. 또한 다른 문서와의 재 포매팅도 가능하게 된다. XML은 여러 형태의 문서 형태를 제공하여 문서의 유통면에서도 많은 이점을 얻을 수 있다. 또한 별도의 작업 없이 사용된 로직과 문서를 사용할 수 있게 된다.

패키지의 수가 실제 사용되는 클래스의 수에 비해서 다소 많은 듯하나 이는 차후 버전을 고려하여 설계하였다. 시스템 설계 패키지 설명은 <표 26>과 같다.

〈표 26〉 시스템 설계 패키지 설명 테이블

패키지명	설명
pe.eunseo	서블릿으로 구성하여 WAS에서 불려지는 모듈들
pe.eunseo.DB	직접적으로 데이터베이스와의 트랜잭션을 관련된 모듈들
pe.eunseo.Mgr	자바서버페이지 레벨에서 불려지는 객체들의 집합
pe.eunseo.Rule	Config 파일을 읽고 필요한 값들을 취득하기 위한 모듈
pe.eunseo.util	XML 파싱 등의 유틸 모듈

객체 간의 시퀀스 다이어그램은 (그림 8)과 같이 만들어졌다.

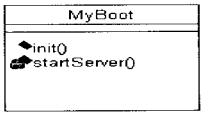


(그림 8) 부팅의 순서

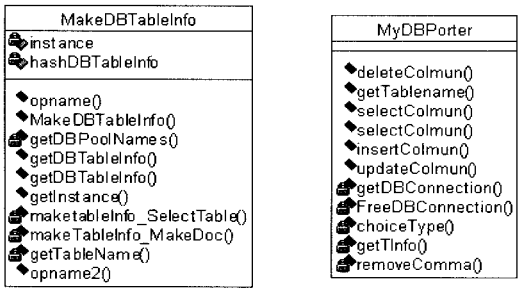
부팅 순서는 접속 풀(Connection Pool)를 생성하고 그 접속 풀을 이용해서 테이블의 정보를 취득한다.

본 구현에서는 WAS의 부팅 순서 정보를 이용하여 사용되는 객체(Object)들을 차례로 로드한다. 로드된 객체(Object)들은 자바 서버 페이지(JSP)등의 프리젠테이션 레벨에서 사용된다.

MakeDBTableInfo는 ConnectionMgr에서 접속(Connection)에 대한 정보를 얻고 이를 근거로 사용가능한 모든 테이블의 정보를 수집한다. 그리고 수집한 정보를 XML 돔 트리(DOM Tree)의 형태로 보관한다. 보관된 돔 트리 객체(DOM Tree Object)를 이용해서 데이터베이스 포터(DBPorter)는 각각의 테이블과의 원활한 트랜잭션(Transaction)을 위한 쿼리를 작성하고 이를 사용해서 데이터베이스에 추가(Insert), 업데이트(Update), 삭제(Delete), 선택(Select)의 기본적인 동작을 수행한다.



(그림 9) MyBoot의 클래스 다이어그램



(그림 10) MakeDBTableInfo와 MyDBPorter의 클래스 다이어그램

MakeDBTableInfo는 테이블에 대한 정보를 정리하고 이를 돔 트리 객체(DOM Tree Object)를 만들고 이를 저장한다. 그리고 이와같은 객체(Obejct)를 이용해서 데이터베이스 포터(DBPorter)는 데이터베이스와의 트랜잭션(Transaction)을 담당한다. 클래스 다이어그램의 속성과 메소드 설명은 <표 27>과 같다.

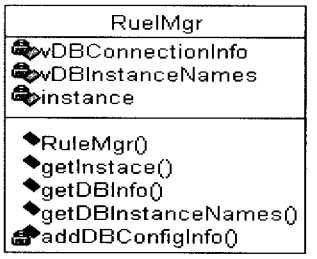
〈표 27〉 MakeDBTableInfo 클래스 다이어그램의 속성과 메소드 설명 테이블

Attributes명	설명
instance	MakeDBTableInfo 자체를 나타내는 것
hashDBTableInfo	관련된 테이블에 대한 정보를 저장하고 있는 객체
Method 명	설명
MakeDBTableInfo	생성자
getDBPoolNames	RuleMgr에서 PoolNames들을 가져옴
getDBTableInfo	모든 정보를 반환함
getDBTableInfo	특정 정보만을 반환함
getInstance	인스턴스를 가져옴
maketableinfo_SelectTable	특정 테이블의 모든 필드와 속성값을 가져옴
maketableinfo_MakeDoc	테이블의 정보를 돔(DOM)의 형태로 만듬
getTableName	테이블의 이름 모두를 가져옴

〈표 28〉 MyDBPorter 클래스 다이어그램의 메소드 설명 테이블

Method 명	설명
deleteColmun	해당 테이블에 대한 트랜잭션 담당
getTablename	
selectColmun	
selectColmun	
insertColmun	
updateColmun	
getDBConnection	Connection 가져오기
FreeDBConnection	Connection 반환하기
choiceType	필드 타입에 따른 스트링 변환을 담당
getTInfo	RuleMgr에서 정보 가져 오기
removeComma	콤마(Comma)제거

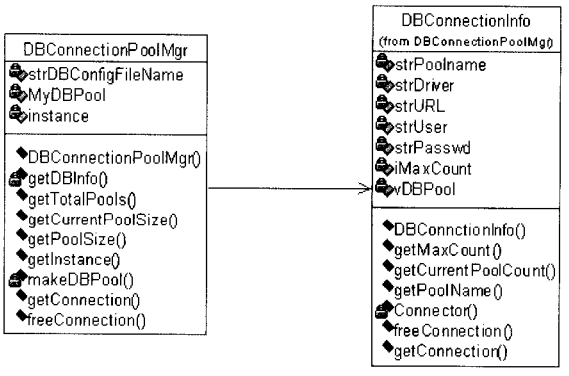
본 구현에서 사용되는 모든 객체(Object)에 대한 기초 자료를 Config 파일에서 읽고 이를 저장한다. (그림 11)의 클래스 다이어그램의 속성과 메소드 설명은 다음 표와 같다.



(그림 11) RuelMgr의 클래스 다이어그램

〈표 29〉 RuelMgr 클래스 다이어그램의 속성과 메소드 설명 테이블

Attribute	설명
vDBConnectionInfo	Connection Pool을 만드는데 필요한 기초 자료의 Collection
vDBInstanceNames	Table에 기초 자료의 Collection
instance	RuleMgr 자체
Method 명	설명
Rulemgr	생성자
getInstance	인스턴스를 가져옴
getDBInfo	Connection에 대한 정보를 제공한다.
getDBInstanceNames	테이블에 대한 정보를 제공한다.
addDBConfigInfo	XML Config를 읽고 사용되는 객체를 생성한다.



(그림 12) DBConnectionPoolMgr과 DBConnectionInfo 클래스의 다이어그램

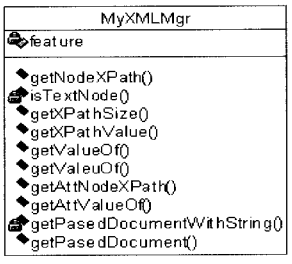
본 구현에서 사용하는 ConnectionPool는 일반적으로 상업용 DBMS에서 제공하는 것과는 틀리다. 일반적으로 가장 많이 사용하는 Hashtable이란 Java의 Object를 사용하였고, 아울러 제공된 Connection을 회수하고 그 Connection을 다시 사용한다. 즉, Config에 Connection의 수 만큼만 만들고 더 이상은 만들지 않는다. (그림 10)의 클래스 다이어그램의 속성과 메소드 설명은 <표 30>과 같다.

〈표 30〉 DBConnectionPoolMgr 클래스의 다이어그램의 속성과 메소드 설명 테이블

Attribute	설명
MyDBPool	ConnectionPool를 담고 있는 객체
instace	
Method명	설명
DBconnectionPoolMgr	생성자
getDBInfo	RuleMgr에서 필요한 정보 취득
getTotalPools	만들어진 풀(Pool)에 대한 크기 제공
getCurrentPoolSize	
getInstance	
makeDBPool	내부 클래스를 이용해서 Connecition을 만듦
getConnection	Connetion 제공
freeConneciton	Connetion 회수

〈표 31〉 DBConnectionInfo 클래스의 다이어그램의 속성과 메소드 설명 테이블

Attribute	설명
strPoolname	풀(Pool)의 이름
strDriver	JDBC Connection을 위해서 필요한 정보
strURL	
strUser	
strPasswd	
iMaxCount	풀(Pool)에 저장될 Connection 수
vDBPool	DBConnectionPool
Method명	설명
DBConnectionInfo	생성자
getMaxCount	총 Connection의 수
getCurrentPoolCount	현재 풀(Pool)이 가지고 있는 Connection 수
Connector	Connection을 맺음
freeConnection	Connection 회수
getConnection	Connection 제공



(그림 13) MyXMLMgr의 클래스 다이어그램

MyXMLMgr Class는 XML 형식으로 되어있는 Config 파일을 파싱하고 필요한 값을 XPath를 이용해서 찾고 이를 반환하는 역할을 담당한다. XML을 파싱하는 방법으로 최근엔 SAX를 많이 사용하고 있으나 본 구현에서는 DOM을 이용하였다. Config정보라는 특수성과 XPath 경로 정보를 이용하므로 구지 SAX와 같은 엔트리(Event) 중심의 파싱보다는 트리와 같은 노드중심의 돔(DOM)이 유리하다.

4.1.2 데이터베이스 설계

일정 관리를 위한 OT 프레임워크의 데이터베이스 내용은 <표 32>와 같다.

〈표 32〉 데이터베이스 필드 테이블-1

Field Name	Type	Comment
MenuID	varchar(2)	메뉴의 아이디, 트리를 그리기 위해서 필요
MenuName	varchar(100)	각 단계의 이름
F1	varchar(100)	각 단계에서 필요한 작업을 위함. [일종의 비고 필드로 존재함]
F2	varchar(100)	
F3	varchar(100)	
F4	varchar(100)	
F5	varchar(100)	
F6	varchar(100)	
F7	varchar(100)	
F8	varchar(100)	

〈표 33〉 데이터베이스 필드 테이블-2

Field Name	Type	Comment
SeqID	int(10)	각 필드의 Key
ParentSeqID	int(10)	수정되었을시에 원본의 SeqID
MenuID	varchar(2)	main_menu의 MainID
ISSubNode	varchar(1)	하위 노드의 존재 유무 Y/N
SubNodeID	int(3)	하위 노드로 존재할시에 부여되는 ID
SubMenuID	varchar(4)	하위 노드로 존재할 시에 상위 노드에 해당되는 ID
SubName	varchar(100)	현 필드에 해당되는 이름.
F1	varchar(100)	Parent 항목
F2	varchar(100)	Children 항목
F3	varchar(100)	Activity 항목
F4	varchar(100)	Solution 항목
F5	varchar(100)	Precondition 항목
F6	varchar(100)	PostCondition 항목
F7	varchar(100)	Contract 항목
F8	varchar(100)	Document 항목

〈표 34〉 데이터베이스 필드 테이블-3

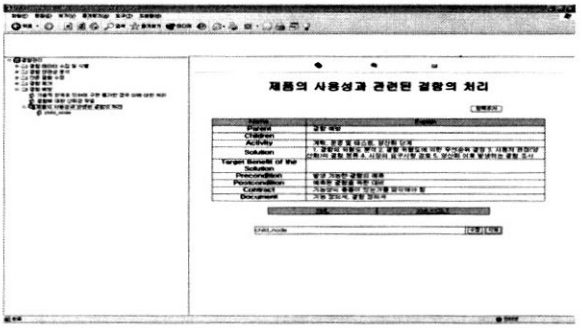
Field Name	Type	Comment
ID	varchar(10)	사용자 ID
Passwd	varchar(10)	사용자 Password
Name	varchar(100)	사용자 이름
Address1	varchar(100)	사용자 주소
Address2	varchar(100)	
Address3	varchar(100)	
email	varchar(100)	사용자 메일 주소
phone1	varchar(100)	사용자 전화 번호
phone2	varchar(100)	
Role	varchar(10)	사용자의 등급 (admin / user)

4.2 화면 설계

로근인 후 OT 프레임 워크의 개요와 하위 항목을 추가, 수정, 삭제할 수 있는 화면은 (그림 14)와 같다.

(그림 14) 항목 추가 기능 및 내용 뷰어(Contents Viewer)

하위 노드의 추가를 위한 정보 제공과 보고를 위한 XML의 제공 화면은 (그림 15)와 같다. 수정과 삭제를 통하여 하위 항목을 제어할 수 있게 된다.



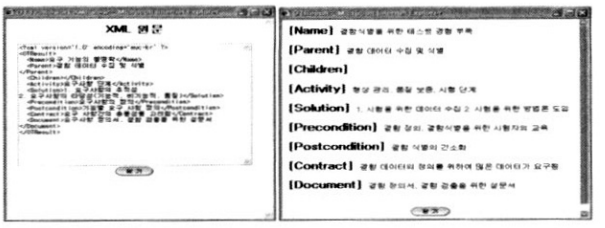
(그림 15) 하위 노드 추가

```
<?xml version="1.0" encoding="euc-kr"?>
<xsl:stylesheet          xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
<xsl:output method="html" encoding="euc-kr"/>

  <xsl:template match="/">
    <xsl:apply-templates select="OTResult"/>
  </xsl:template>

    <xsl:template match="OTResult">
      <html>
      <head>
          <title>OTResult</title>
    </head>
    <body>
      <p><b><font          size="5">[Name]          </font></b><xsl:value-of
select="Name"/></p>
      <p><b><font          size="5">[Parent]        </font></b><xsl:value-of
select="Parent"/></p>
      <p><b><font          size="5">[Children]      </font></b><xsl:value-of
select="Children"/></p>
          <p><b><font      size="5">[Activity]      </font></b><xsl:value-of
select="Activity"/></p>
          <p><b><font      size="5">[Solution]      </font></b><xsl:value-of
select="Solution"/></p>
      <p><b><font          size="5">[Precondition]  </font></b><xsl:value-of
select="Precondition"/></p>
          <p><b><font      size="5">[Postcondition] </font></b><xsl:value-of
select="Postcondition"/></p>
          <p><b><font      size="5">[Contract]      </font></b><xsl:value-of
select="Contract"/></p>
          <p><b><font      size="5">[Document]      </font></b><xsl:value-of
select="Document"/></p>
          </body>
          </html>
    </xsl:template>
      <xsl:template match="Name">
      <xsl:apply-templates select="*|text()"/>
    </xsl:template>
</xsl:stylesheet>
```

(그림 16) 보고서 생성을 위한 XML 소스 코드



(그림 17) 웹 브라우저에서의 XML 형태

5. 결론 및 향후 연구 방향

본 논문에서는 결함 처리를 위하여 프로젝트 관리자와 품질 담당자에게 지침을 제공하여 문제 발생 시, 빠른 해결책을 제시하고, 이를 기반으로 하여 유사한 프로젝트 수행 시 문제를 예측 및 예방할 수 있는 방향을 제시하였다. 일정관리를 목표로 한 OT를 사용함으로서 결함 발생 시, 해당되는 항목을 검색하여 해결책을 찾아서 빠른 시간 내에 조치를 취할 수 있게 된다.

일정 관리 위한 OT 프레임워크는 웹 서비스를 지원하기 위하여 구현되었으며, 일정 관리를 위한 활동과 해결책을 연구하여 지속적으로 내용이 업데이트가 되도록 설계되었다. 이를 활용하여 관리자는 일정 관리를 수행하고 결국 프로젝트 관리의 성숙도 수준을 높일 수 있는 개선 작업을 병행하게 된다. 따라서 일정 관리를 위한 OT 프레임워크를 활용함으로써 다음과 같은 프로세스 개선 효과와 조직의 비전 달성을 위한 측정 도구로 활용할 수 있다.

향후 연구로는 소프트웨어 프로세스 개선과 관련된 중요 요인인 비용에 관한 OT 설계를 수행하며, 사례연구의 자료를 30개 이상 수집하여 통계적인 분석을 수행하는 것이다. 이때 설문서 조사 및 분석에 의한 지연요소는 Agile 방법론을 적용하여 간소화 시키는 방안을 연구하겠다. 결과적으로 소프트웨어 프로세스 개선이라는 OT 프레임워크를 설계를 완성하는 것이 향후 연구 계획으로 제시할 수 있다.

참 고 문 헌

[1] 이경환, "HDC를 위한 모델링", 제 5회 한국정보과학회 소프트웨어공학 연구회, 2003.01.

[2] Annual research review, USC/CSE workshop reports, 2002.

[3] Software process improvement forum, KASPA SPI-7, 2002.

[4] George eckes, "The six sigma revolution", John willey & sons, 2001.

[5] SPICE assessment in korea, The korea SPICE, 2002.

[6] 신경애, "결함중요도 단계를 고려한 소프트웨어 신뢰도 성장 모델에 관한 연구", 컴퓨터 산업교육기술 학회논문지, Vol. 3, No.07 pp.837-844, 2000.

[7] Robert h. dunn, "Software defect removal", Mcgraw-hill, 1984.

[8] Ram chillarrege, Kothanda r. prasad, "Test and development

process retrospective a case study using ODC triggers", IEEE computer society, 2002.

[9] N. fenton and N. ohlsson, "Quantitative analysis of faults and failures in a complex software system", IEEE Trans. Software Eng, pp.797-814, 2000.

[10] McCall, P.K. richards and G.F. walters, "Factors in software quality", Vol.1, 2 and 3. Springfield VA, AD/A-049-014/015/055, 1997.

[11] Vouk, Mladen, "Software reliability engineering, Tutorial notes topics in reliability & maintainability & statistics", 2000 annual reliability and maintainability symposium, 2000.

[12] Padberg, "A fast algorithm to component maximum likelihood estimates for the hypergeometric software reliability model", Asia-pacific conference on quality software APAQS, 2001.

[13] Vamder wiel, Votta, "Assessing software designs using capture-recapture methods", IEEE transaction on software engineering Vol.2 pp.1045-1054, 1993.

[14] Wohlin, Runeson, "Defect content estimations from review data", Proceedings international conference on software engineering ICSE pp.400-409, 1998.

[15] Gaffney, John, "Some models for software defect analysis", Lockheed martin, 1996.

[16] L.hatton, "Is modularization always good idea", Information and software technology, Vol.38, pp.719-721. 1996.

[17] B. compton and C. withrow, "Prediction and control of ada software defects", J. systems and software, Vol.12, pp.199-207, 1990.

[18] N. fenton and M. neil, "Software metrics : successes, failures, and new directions", J. systems and software, Vol. 47, pp.149-157, 1999.

[19] Roger s. pressman "Software engineering" Mcgraw-hill international edition, 1997.

[20] Tim kasse, "Actin focused assessment for software process improvement", Artech house, 2002.

[21] Thomas saaty, "Analytical planning", Rws publications, 1985.

[22] Victor basili, G. caldiera and D. rombach, "The experience factory", Encyclopedia of software engineering, Wiley, 1994.

[23] Victor basili, G. caldiera and D. rombach, "The goal question metric approach", Wiley, 1994.

[24] Victor basili, "The experience factory and its relationship to other improvement paradigms", Lecture notes in computer science, 1993.

[25] Barry boehm, "Software cost estimation with COCOMO II", Prentice-Hall PTR, 2000.

[26] Paulish, and Carleton, "Case studies of software process improvement measurement," Computer, Vol.27, No.9, 1994.

[27] Roger s. pressman, "Software engineering: A practitioner's approach", Mcrwa-hill international editions, 1997.

[28] 이은서, 이경환, "소프트웨어 결함 분석을 위한 트리거 설계", 한국정보처리학회논문지, 2003.

[29] 이은서, 이경환, 소프트웨어 결함 처리를 위한 Opportunity Tree 및 알고리즘 설계, 제11-D권 제4호 PP873, 한국정보처리학회 논문지, 2003. 08.

[30] Chris frye, "Understanding components", Andersen consulting knowledge exchange, 1998.

[31] Grady booch, "A primary contributor to the UML and rational software's objectory process", 1997.

[32] Tim kasse, "Actin focused assessment for software process improvement", Artech house, 2002.

이 은 서

e-mail : eslee1@ssu.ac.kr

200년~현재 ISO/IEC 15504 국제 심사원

2004년 중앙대학교 컴퓨터공학과(박사)

2004년~현재 임베디드 산업협회 전문 위원

2004년~현재 한국정보통신기술협회 위원

2005년~현재 숭실대학교 정보미디어기술 연구소 연구교수

관심분야 : CBD, Formal method, Quality model, SPI(Defect Analysis)

이 상 호

e-mail : shlee@computing.ssu.ac.kr

1984년 서울대학교 컴퓨터공학과(학사)

1986년 미국 노스웨스턴 대학교 전산학과 (석사)

1989년 미국 노스웨스턴 대학교 전산학과 (박사)

1990년~1992년 한국전자통신연구원 선임연구원

1992년~현재 숭실대학교 컴퓨터학부 교수

1999년~2000년 미국 George mason 대학교 교환 교수

관심분야 : 인터넷 데이타베이스, 데이타베이스 튜닝 및 성능평가, 웹 기술