소프트웨어 결함 처리를 위한 Opportunity Tree 및 알고리즘 설계

이 은 서[†] · 이 경 환[††]

요 약

본 연구에서는 소프트웨어 개발 시 발생하는 결함을 찾아내고, 원인을 식별 및 해결책을 제시하고자 한다. 또한 검출된 결함 항목을 기반으로 하여 결함간의 연관성을 파악하여 opportunity tree로 나타낸다. 신뢰성 있는 소프트웨어를 개발하기 위해서는 소프트웨어와 개발과정에 존재하는 결함을 찾아내고 이를 관리하는 것이 중요한 요인이 된다. 이와 같은 요인은 품질로 귀결되게 되는데, 품질은 비용, 일정과 함께 프로젝트의 성공을 결정하는 주요 요소이다. 따라서 결함 처리 opportunity tree 및 알고리즘을 이용하여 유사한 프로젝트를 수행 시, 결함 예측하여 대비할 수 있게 된다.

Design of Software Opportunity Tree and Its Algorithm Design to Defect Management

Eun Ser Lee[†] · Kyung Whan Lee[††]

ABSTRACT

This research provides the solution of defect problem and detection of defect and its causes that happen on software development. For developing a reliable software, a key factor is to find and manage defects that are during software development. Based on defect items analysis, we understand associated relation between defects and design defect opportunity tree. Developing the similar project, we can estimate defect and prepare to solve defect by using defect management opportunity tree.

키워드 : 소프트웨어 프로세스 개선(SPI), 결함 관리(Defect Management), 결함 분석(Defect Analysis), 결함 트리거(Defect Trigger), 프로젝트 관리(Project Management), 결함 관리 OT(Opportunity Tree)

1. 연구 배경

소프트웨어 공학의 가장 중요한 목표는 고품질의 시스템과 소프트웨어 또는 제품을 생산해 내는 것이다. 이 목표를 달성하기 위해 소프트웨어 엔지니어는 소프트웨어 프로세스의 전후 관계를 분석하기 위햇 새로운 도구들과 연관된 효율적인 방법을 적용해야 한다. 컴퓨터 시스템의 고장은 하드웨어적인 것 보다 소프트웨어적인 원인이 더 많이 있다[1]. 따라서 소프트웨어적인 원인을 찾고 개발하기 위하여 소프트웨어 공학기술을 이용하여 신뢰성이 높은 소프트웨어를 개발하는 것이 중요한 부분이다.

그러므로 소프트웨어 공학에서 품질이라는 것이 중요한 요인이 되었다.

시스템, 소프트웨어 또는 제품의 품질은 문제를 기술하는 요구사항, 해결 방안을 모형화시키는 설계, 실행 가능한 프로그램을 이끌어 내는 코드, 소프트웨어의 오류를 발견하기 위해 소프트웨어를 조사하는 테스트 등을 기술하는데 아주 좋다[6]. 훌륭한 소프트웨어 엔지니어는 분석과 설계 모형의 품질, 원시코드, 그리고 실험적으로 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, 6].

품질은 프로젝트 관점에서도 살펴본다면 다음과 같다. 프로젝트 관리자는 프로젝트가 진행되면 품질을 평가해야 한다. 개별적으로 소프트웨어 엔지니어가 수집한 비공개 메트릭스는 프로젝트 수준의 결과들을 제공하기 위해 통합된다. 비록 많은 품질 측정이 수집될 수 있어도, 프로젝트 수준의 결과들을 제공하기 위해 통합되어야 한다[3]. 프로젝트 수준에서 오류와 결함을 측정하는 기준이 된다. 이들 측정으로부터 유도된 메트릭스는 개인과 그룹 소프트웨어의 품질 보증과 제어 활동의 효과성에 대한 지표를 제공해 준다[4].

※ 이 논문은 2003년도 중앙대학교 학술 연구비 지원에 의한 것임.
† 준 회 원 : 중앙대학교 대학원 컴퓨터공학과
†† 정 회 원 : ISO/SC7/WG10 SPICE 한국 위원장
논문접수 : 2004년 5월 4일, 심사완료 : 2004년 5월 28일

검토 기간에 발견된 오류와 테스팅 기간에 발견된 오류는 메트릭스가 갖고 있는 각 활동의 효율성에 대한 통찰력을 제공해준다. 오류 데이터는 또한 각 프로세스 프레임워크 활동에 대한 결함 제거 효율성(DRE : Defect Removal Efficiency)을 계산 하는데 사용될 수 있다.

신뢰성 있는 소프트웨어를 개발하기 위해서는 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 정의 할 수 있는 것이 아니다. 소프트웨어는 다양한 품질 관련 특성을 가지고 있는데 품질을 위한 국제 표준도 있다[2].

실제로 품질 관리는 결함의 관점에서평가하게 되므로 인도된 결함 밀도, 즉 인도된 소프트웨어의 결함의 개수/단위 크기를 품질의 정의로 사용하는데, 이것은 사실 현재 업계에서 표준 정의가 되어가고 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 결함 처리를 위하여 중요한 항목을 식별하고 연관성을 분석하게 된다. 또한 이를 기반으로 결함 관리를 위한 opportynity tree(이하 OT)를 설계하고자 한다.

2. 기본 개념

2.1 결 함

결함 정보는 다른 유형의 원시데이터로서, 소프트웨어 프로젝트에서 매우 중요한 요소이다. 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 관한 정보는 여러 가지 다양한 결함 검출 활동을 통해 발견한 결함의 개수도 포함 한다. 그래서 요구사항 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 능력 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 결함의 영향

발견된 결함이 기록되면 발견된 결함의 개수, 시작 상태의 결함의 비율, 그리고 프로젝트의 다른 쟁점 등을 집중적으로 분석할 수 있다. 이와 같은 결함 추적은 프로젝트 관리를 위한 최선의 실행 지침중의 하나이다[8, 9].

필요한 다른 유형으로 분석하려고 할 때 단순히 결함을 기록하는 것만으로는 충분하지 않다. 예를 들어 사용중인 프로세스의 품질 역량을 이해하려면, 테스트 중 발견한 결함과 인도 후에 발견한 결함을 결함의 다른 유형들과 분리해야 한다. 몇 퍼센트의 결함이 어디서 발견되었는지를 기록할 기준선을 만들기 위해, 그리고 실제 발견한 결함과 예측한 결함을 비교하기 위해 결함을 발견한 단계를 식별하면, 결함의 영향 분석도 가능해진다[17].

그러나 이런 정보만으로는 여전히 충분하지 못하다. 조직의 목표 중에 하나는 지속적인 프로세스를 개선해서 품질과 생산성을 개선하는 것이다. 품질과 생산성을 개선하는 접근 방법은 다양한 결함 검출 단계의 결함 제거 효율에 대해 연구하고, 어디에 개선의 여지가 있는지 알아내는 것이다. 결함 검출 단계의 결함 제거 효율은 단계를 수행할 때 나타난 결함의 전체 개수에 대한 그 단계에서 발견된 결함 개수의 비율이다. 결함 제거 효율을 높이면, 결함의 잠복 가능성이 낮다[3]. 이때 분명한 것은 생산성을 개선하기 위한 방법이 결함 제거 효율개선이라는 것이다. 결함 제거 효율을 계산하려면, 결함을 발견한 곳과 결함이 주입된 시점을 알아야 한다. 다시 말해서 기록되는 각 결함에 대해서 그 결함이 시스템에 주입된 단계에 대한 정보도 파악해야 한다.

인포시스에서는 상용 결함 통제 시스템을 결함의 기록과 추적을 위해 프로젝트에서 사용했다[20]. 결함 통제 시스템은 다양한 유형의 분석을 제공하며, 결함분석에도 사용한다. 또한 각 프로젝트에 대해 결함 통제 시스템을 설정하고, 모든 프로젝트 구성원이 프로젝트를 위한 결함 통제 시스템 데이터에 접근하는 것을 허용한다. 결함 통제 시스템을 설정하는 중에는 프로젝트에서 기록할 결함에 대한 정보의 유형을 상세히 기술하게 된다. 즉 필요에 따라서 프로젝트에 기록할 정보를 최적화한다. 인포시스는 모든 프로젝트에 필요한 몇 개의 필드를 필수 항목으

로 명세 했다. <표 1>에서는 다양한 선택 또는 필수 항목을 제시하고 있다.

〈표 1〉 결함 데이터

데이터	데이터의 설명	필수/선택
프로젝트 코드	결함이 발견된 프로젝트 코드	필 수
설 명	결함에 대한 설명	필 수
모듈 코드	모듈 코드	선 택
프로그램명	결함이 발견된 프로그램명	선 택
검출된 단계	결함이 검출된 단계	필 수
주입된 단계	결함이 주입된 / 출처 단계	필 수
유 형	결함의 분류	필 수
심각도	결함의 심각도	필 수
검토 유형	검토 유형	선 택
상 태	결함의 현재 상태	필 수
제출자	결함을 발견한 사람 이름	필 수
소유자	결함을 소유한 사람의 이름	필 수
제출일	결함이 소유자에게 제출된 날짜	필 수
마감일	제출된 결함이 마감된 날짜	필 수

정량적인 품질관리를 위한 접근방법은 결함 예측을 이용하는 것이다. 이 접근 방법에서 인도된 결함 밀도의 관점에서 품질 목표가 설정된다. 품질 관리의 다른 측면, 즉 개발 프로세스 관리는 프로세스가 원하는 결함 밀도 목표를 달성할 수 있는 정량적인 방법으로 통제되어야 한다. 이 목표는 다양한 결함을 검출하는 활동에 의해 식별될 결함 예측, 그리고 실제 결함 개수와 예측된 결함 수준 비교 등을 통해 이루어진다. 일단 품질 목표를 설정했으면, 서로 다른 단계의 결함 수준을 예측하고, 예측을 맞춤으로써 목표 품질을 달성하게 된다. 이때 프로세스를 관리하기 위해 예측된 결함 수준은 실제 결함 수준과 비교하는 기준이 된다. 이것을 통해 개발 프로세스가 품질 목표를 달성하기 위한 방향으로 움직였는지의 여부를 평가하게 된다.

2.3 결함 처리 항목 분석

결함 처리 항목 분석은 위험한 사건 또는 시스템 상태를 이끌어 낼 수 있는 사건들의 순차적이고 동시 결합인 그래픽 모형을 만들어 준다. 잘 개발된 결함 처리 방법을 사용하면 다른 시스템 요소들에서 발생하는 일련의 상호 관련된 결함의 결과를 관측할 수 있다. 실시간 논리는 지정된 사건들과 대응하는 행동들로 시스템 모형을 만든다. 사건-행동 모형은 시스템 요소들과 그들의 시간에 관한 안전성 주장들을 시험하려고 논리 연산을 사용해서 분석한다. 페트리 넷 모형은 가장 위험한 결함을 발견하는 데 사용될 수 있다[19].

위험이 식별되고 분석되었으면, 안전성에 관련된 요구사항이 소프트웨어에 명시될 수 있다. 즉, 명세는 바라지 않는 사건들의 목록과 희망하는 시스템 응답을 포함시킬 수 있다. 희망하지 않는 사건을 관리하는 소프트웨어의 역할은 이때 지적된다.

소프트웨어 신뢰성과 소프트웨어 안전성이 아주 밀접한 관계를 갖고 있다해도 그들간의 미묘한 차이를 이해하는 것은 중요하다. 소프트웨어 신뢰성은 소프트웨어 결함이 발생할 수 있는 확률을 구하기 위해서 통계적 분석을 사용한다. 그러나 결함의 발생은 필연적으로 위험이나 재난으로 귀결되지 않는다[20].

소프트웨어 안전성은 실패가 재난을 유발시킬 수 있는 조건에서 귀결되는 방법으로 시험한다. 즉, 실패는 공허한 상태로 고려되지 않고 전체 컴퓨터-기저 시스템에서 평가된다.

3. 결함 처리를 위한 알고리즘

본 장에서는 회사에서 실제 프로젝트를 수행하는 과정에서 발생하는 결함을 해결하기 위한 OT를 제공하고자 한다. 또한 결함 처리를 위한 OT 설계 알고리즘을 제안하고자 한다. 이를 위하여 결함 처리를 위하여 요구되어 지는 항목을 찾아야 한다. 따라서 결함 처리를 위하여 요구되는 항목을 찾기 위하여, 설문서를 작성하게 된다. 또한 결함을 검출하기 위한 설문서의 구조와 내용을 설명하고, 결과를 계층적 의사 결정법에 의하여 분석하게 된다. 그리고 분석된 결과를 결함 처리 방안을 시스템화하기 위하여 결함 처리 OT를 설계한다.

설문서 대상 회사 및 프로젝트의 도메인으로는 디지털 시스템을 대상으로 수행하였으며, 네 개회사의 21개 프로젝트를 대상으로 하였다. 설문서 조사 및 분석 방법은 GQM (Goal-Qustion-Metric) 방법을 채용하였다.

3.1 결함 관리를 위한 OT 프레임워크의 개념 및 필요성

결함 처리 OT는 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 요청하기 위하여 질의응답 게시판을 이용하여 질문을 하게 되고, 그 결과를 리턴 받기 위하여 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 결함 관리를 하기 위해서는 결함 관리 항목을 정해서 지속적으로 항목의 변경 및 추가 사항이 존재하는가를 검사해야 한다. 이를 통하여 정확하고 신속한 결함 관리를 수행할 수 있게 된다.

결함 관리를 수행하지 않았을 경우, 다음과 같은 문제점을 발생시킬 수 있다.

- 프로젝트 수행 시, 결함을 해결하기 위한 추가 비용의 발생
- 프로젝트 수행 시. 결함에 의한 일정 지연 발생
- 프로젝트 수행 시, 결함에 의한 제품의 품질 저하
- 결함 발생 시, 이를 해결하기 위한 방법 부재
- 결함 발생 시, 원인 파악의 난해성
- 유사한 프로젝트 수행 시, 동일한 내용의 결함 발생

결함 관리를 하지 않았을 경우, 위에서 제시한 내용의 문제점이 발생될 수 있다. 따라서 문제점들을 완화시키기 위하여 결함 처리 OT를 제안하고자 한다.

3.2 결함 처리를 위한 OT 알고리즘

결함 처리를 위하여 GQM 방식에 의한 OT 알고리즘을 기반으로 하여 다음과 같이 제시한다.

단계 1 : 찾고자 하는 목표 선정(비즈니스 목표 결정)
단계 2 : 목표를 달성하기 위한 하위 목표 선정
단계 3 : 목표와 하위 목표의 타당성 및 신뢰성 검증을 위 한 설문서 작성
단계 4 : 목표와 하위 목표를 결정하기 위한 의사 결정 (계층적 의사 결정법의 적용)
단계 5 : 목표와 하위 목표 선정후 해결책 제시
단계 6 : 선정된 항목을 OT로 구현

본 논문에서는 위에서 제시한 알고리즘을 기반으로 하여 결함 처리 OT를 설계하였다. 설계 내용은 다음과 같다.

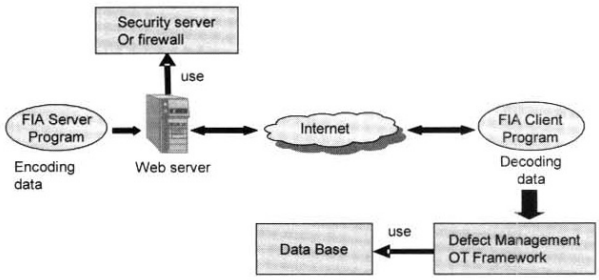
3.3 시스템 환경 분석

3.3.1 시스템 구성인자 식별

결함 처리 OT를 설계하기 위해서는 선행작업으로서 시스템 환경의 분석이 요구된다. 시스템 환경은 결함 처리 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 요구 사항의 충돌 및 실현가능성을 판별할 수 있다. 그러므로 시스템 구조도에서는 시스템 요소간의 관계를 명시함으로서, 연관성을 쉽게 인식할 수 있다.

본 논문에서는 결함 처리 OT를 설계하기 위한 관점으로 시스템 환경을 분석하였다. 시스템 구성인자는 결함 처리 OT를 구축하기 위하여 요구되는 하드웨어를 식별하는 것이다. 따라서 식별된 하드웨어 환경 하에 결함 처리 OT가 구축될 수 있게 된다. 시스템 구성인자로는 보안을 처리할 서버와 사용자가 결함 처리 OT를 접근하여 사용할 수 있게 해주는 웹 서버, 그리고 결함 처리 OT에서 자료를 참조하는 데이터 베이스를 구성인자로 추출하였다. 식별된 시스템 구성인자는 (그림 2)와 같다.



(그림 1) 시스템 구조도

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 결함 처리 OT를 설계하는 것이 목적이며, 이를 위하여 다음과 같은 스테이크 홀더를 추출할 수 있었다. 추출된 스테이크 홀더의 정의는 다음과 같다.

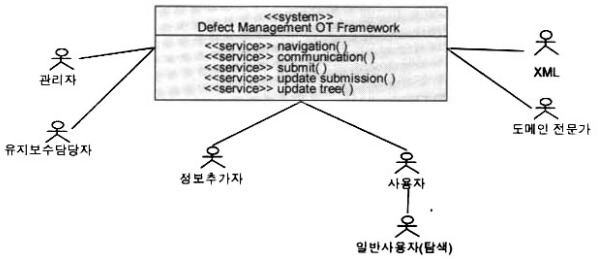
〈표 2〉 스테이크 홀더의 정의

스테이크 홀더 명	설 명
설문서 작성자	OT를 개발하기 위하여 설문서를 작성하여 초기 요구사항을 조사하는 사람
개발자	OT를 개발하는 사람
결함 처리 OT 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 유지 보수 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 결함 처리 OT의 시나리오를 추출할 수 있다. 추출된 시나리오

는 요구사항이 기능화하기 위하여 필수적으로 요구되는 것으로서, 기능들 간의 실현 가능 타당성을 제시할 수 있게 된다. 각 스테이크 홀더의 연관관계는 다음과 같다.



(그림 3) 스테이크 홀더 구조도

3.3 요구사항 추출

요구사항 분석을 위한 설문서는 GQM 방식에 의하여 설문서를 작성하였으며, 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 서브 목표는 전체 목표를 달성하기 위한 단위로 간주 할 수 있다.

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

결함 처리 OT 요구사항 추출 설문서는 다음과 같다.

〈표 3〉 결함 처리 OT 요구사항 추출 설문서

결함 관리 OT 프레임워크 요구사항 추출 설문서	
일반적인 사항	1. 작성 일자 :
	2. 작성자 직급 :
	3. 프로젝트명 :
	4. 프로젝트 특성 분류 :
결함 관리의 이슈사항	1. 이전과 현재의 프로젝트 수행 경험에 기반하여 결함 중에서 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. 응답은 전체 152개를 수행하였으며, 오류 응답은 존재하지 않았다. 빈도를 상세화 하면 다음과 같다.

분석된 결함 처리 OT 항목 빈도에 의하면 여러 항목 중에서 마케팅 전략을 제외한 나머지 항목에 대하여 필요성이 있다고 응답을 했다. 따라서 마케팅 전략을 제외한 이외의 항목이 결함 처리 OT의 항목이 대상이 되었다. 그러므로 이후의 결함 처리 OT의 목표는 분석된 항목을 대상으로 수행하게 된다.

<표 4>에서 목표를 결정하였다. 그리고 각각의 목표를 실현하기 위하여 어떤 문제점들이 발생하는지 파악할 필요성이 있다. 또한 식별된 문제점들은 결함 처리 OT를 위한 목표 중에서 어느 범주에 속하는지 구별을 하였다.

〈표 4〉 결함 처리 OT 항목 빈도 분포 표

	Frequency	Percent
결함 자료 수집 및 식별	53	34.8
결함 연관성 분석	32	21.1
잔존 결함 수정	26	17.1
결함 제거	20	13.2
결함 예방	18	11.8
마케팅전략	3	2.0
Total	152	100.0

각각의 결함 처리 OT 항목 내용은 다음과 같이 설명된다.

〈표 5〉 결함 처리 OT 항목의 내용

결함 처리 OT	내 용
결함 자료 수집	결함 관리를 위하여 요구되는 자료를 수집하고 각 개발 단계별 결함을 찾아냄
결함 연관성 분석	각 단계의 결함이 다른 단계의 결함과 연관성이 있는지를 찾아냄
잔존 결함 수정	총 잔존 결함에 대해서 수정 가능한 시간 예측 가능
결함 제거	찾아낸 결함을 제거함
결함 예방	이전의 결함 자료를 이용하여 새로운 프로젝트에서 결함을 예방함
마케팅전략	결함 잔존에도 불구하고 출시해야 하는 경우 마케팅 전략

〈표 6〉 프로젝트 수행 시 발생되는 문제점들

빈 도 / 결함내용	Frequency	Percent
요구기능 불명확	11	5.9
불필요한 결함 자료 작성	22	11.7
결함식별을 위한 테스트 경험 부족	11	5.9
결함 발생 시 명확한 SPEC이 없는 경우 이에 대한 처리	15	8.0
결함의 수정 시 모듈 및 관련문서에 대한 프로세스 및 도구지원	22	11.7
기술적 한계로 인하여 구현 불가한 경우 이에 대한 처리	11	5.9
결함발생으로 인하여 변경사항 관리 방안	11	5.9
결함에 대한 신뢰성 모델	35	18.5
결함 잔존 시 Release 전략	6	3.1
제품의 사용성과 관련된 결함의 처리	15	8.0
결함의 원인에 대한 책임 전가	21	11.2
Spec의 불명확, 미정의에 의한 시험자 주관에 기인한 결함등록	8	4.2
Total	188	100.0

우선 프로젝트 수행 시 발생되는 문제점들은 <표 5>에 제시하였으며, 빈도분석도 병행하여 수행하였다. 분석 결과는 여러 문제점들 중에서 결함을 찾고 해결하기 위한 신뢰성 모델의 필요성이 21.1%를 차지하였고, 불필요한 결함

자료 작성시 소요되는 일정과 비용의 낭비가 12.5%, 결함 발생 시 작업의 책임과 권한이 정확히 명시되지 않아서 결함의 원인에 대한 책임 전가 문제가 11.8%로 많은 비중을 차지하고 있었다.

<표 6>에서 제시된 문제점들은 <표 5>에서 어느 범주에 속하는지 찾아서 연결을 하였다. 이와 같은 작업은 각각의 결함 처리 항목을 완성하기 위한 중요한 요소가 된다. 따라서 결함 처리 OT 항목과 결함 내용의 연관성은 다음과 같이 제시하였다.

〈표 7〉 결함 항목에 따른 결함 내용의 분류표

결함 처리 OT 항목	결함 내용
결함 자료 수집 및 식별	1. 요구 기능의 불명확
결함 자료 수집 및 식별	2. 불필요한 테스트 항목에 의한 결함 식별기간 증가
결함 자료 수집 및 식별	3. 결함식별을 위한 테스트 경험 부족
결함 연관성 분석	4. 결함 발생 시 명확한 SPEC이 없는 경우 이에 대한 처리
잔존 결함 수정	5. 결함의 수정 시 모듈 및 관련문서에 대한 프로세스 및 도구지원
결함 자료 수집 및 식별 결함 예방	6. 기술적 한계로 인하여 구현 불가한 경우 이에 대한 처리
결함 연관성 분석 결함 제거	7. 결함발생으로 인하여 변경사항 관리 방안
결함 식별 결함 제거 결함 예방	8. 결함에 대한 신뢰성 모델
마케팅 전략	9. 결함 잔존 시, Release 전략
잔존 결함 수정 결함 제거 결함 예방	10. 제품의 사용성과 관련된 결함의 처리
결함 연관성 분석	11. 결함의 원인에 대한 책임 전가
잔존 결함 수정	12. Spec의 불명확, 미정의에 의한 시험자 주관에 기인한 결함등록

<표 7>의 내용은 이후의 개발 단계에서 결함 처리 OT 항목 구현 시 완성되어야 할 핵심 내용이 된다. 분석된 하위 목표에서 마케팅이 제외된 이유는 빈도가 낮게 분석되어서 제외되었다. 분석된 하위 목표는 다음과 같다.

〈표 8〉 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

3.4 결함 처리 OT 설계

결함 관리 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 결함 관리 OT 프레임워크의 구성을 트리형태로 설계하였다. 또한 각 OT 항목을 상세히 설명하고자 다음과 같이 구성하였다.

〈표 9〉 결함 관리를 위한 OT 프레임워크 항목

항목 이름	내 용
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

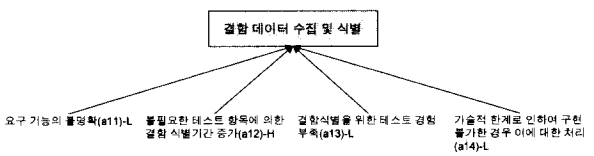
3.4.1 결함 자료 수집 및 식별

결함 자료 수집 및 식별에서 발생하는 결함 내용과 해결책은 다음과 같다. 등급은 빈도에 의한 내용으로 다음과 같이 제시하였다.

〈표 10〉 결함 자료 수집 및 식별 항목의 내용

결 함 내 용	등급
1. 요구 기능의 불명확	Low
2. 불필요한 테스트 항목에 의한 결함 식별기간 증가	High
3. 결함식별을 위한 테스트 경험 부족	Low
4. 기술적 한계로 인하여 구현 불가한 경우 이에 대한 처리	Low

결함 자료 수집 및 식별에 관한 의사결정 구조도와 등급화된 항목은 다음과 같다.



(그림 4) 결함 자료 수집 및 식별의 결함 원인 등급 구조도

항목간 비교 메트릭스는 다음과 같다.

〈표 11〉 결함 자료 수집 및 식별 항목간 비교 메트릭스

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

결함 자료 수집 및 식별 항목간 비교 메트릭스는 a11, a12, a13, a14 열을 기준으로 다른 항목과 중요도를 비교하게 된다. 따라서 a11과 a12는 a11이 3배 많게 분석이 되었다. a11은 L 이고, a12는 H이므로 두 항목간의 등급 차이가 2이므로 중요도 차이 비율은 3배가 된다. a11과 a13은 등급이 L로 같으므로 1배차이가 나게 된다. 그리고 그 이외의 항목들도 같은 방식으로 산출되었다.

결함 자료 수집 및 식별을 수행하기 위한 목표 중 어떤 목표가 중요한가를 산출하여야 한다. 따라서 이와 같은 중요한 목표를 산출하기 위하여 기하 평균을 이용하였다.

결함 자료 수집 및 식별 항목의 목표 중요도는 다음과 같이 산출되었다.

〈표 12〉 결함 자료 수집 및 식별의 하위 목표 중요도 산출표

항 목	산 출 값
a11	$(1 \times 1/3 \times 1 \times 1)^{1/4} = (1/3)^{1/4} = 0.76$
a12	$(3 \times 1 \times 1 \times 1)^{1/4} = 3^{1/4} =$ **1.32**
a13	$(1 \times 1/3 \times 1 \times 1)^{1/4} = (1/3)^{1/4} = 0.76$
a14	$(1 \times 1/3 \times 1 \times 1)^{1/4} = (1/3)^{1/4} = 0.76$

a12(불필요한 테스트 항목에 의한 결함 식별기간 증가) 하위 목표가 결함 자료 수집 및 식별에서 가장 중요한 달성 목표로 분석되었다. 따라서 a12(불필요한 테스트 항목에 의한 결함 식별기간 증가)가 결함 자료 수집 및 식별 목표를 달성하기 위하여 우선순위가 가장 높고, 달성되어야 할 하위목표로 나타났다. 그리고 나머지 a11(요구 기능의 불명확), a13(결함식별을 위한 테스트 경험 부족), a14(기술적 한계로 인하여 구현 불가한 경우 이에 대한 처리)는 같은 중요도를 나타냈으며, a11(요구 기능의 불명확), a12(불필요한 테스트 항목에 의한 결함 식별기간 증가), a13(결함식별을 위한 테스트 경험 부족), a14(기술적 한계로 인하여 구현 불가한 경우 이에 대한 처리)의 목표를 달성하기 위해서는 다음과 같은 해결책을 제시할 수 있다.

〈표 13〉 결함 자료 수집 및 식별 목표를 위한 해결책

결함 내용	해 결 책
1. 요구 기능의 불명확	1. 요구사항의 추적성 2. 요구사항의 타당성(기능적, 비기능적, 품질)
2. 불필요한 테스트 항목에 의한 결함 식별기간 증가	1. 요구사항의 우선순위 식별 2. 요구사항의 추적성
3. 결함식별을 위한 테스트 경험 부족	1. 시험을 위한 자료 수집 2. 시험을 위한 방법론 도입
4. 기술적 한계로 인하여 구현 불가한 경우 이에 대한 처리	1. 요구사항의 타당성(기능적, 비기능적, 품질) 2. 요구사항의 충돌성 판별

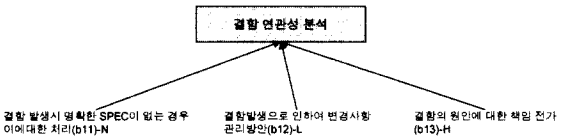
3.4.2 결함 연관성 분석

결함 연관성 분석에서 발생하는 결함 내용과 해결책은 다음과 같다. 등급은 빈도에 관한 내용으로 다음과 같이 제시하였다.

〈표 14〉 결함 연관성 항목의 내용

결함 내용	등 급
1. 결함 발생 시 명확한 SPEC이 없는 경우 이에 대한 처리	Nominal
2. 결함발생으로 인하여 변경사항 관리 방안	Low
3. 결함의 원인에 대한 책임 전가	High

결함 연관성 분석에 관한 의사결정 구조도와 등급화된 항목은 다음과 같다.



(그림 5) 결함 연관성 분석의 결함 원인 등급 구조도

항목간 비교 메트릭스는 다음과 같다.

〈표 15〉 결함 연관성 분석 항목간 비교 메트릭스

	b11	b12	b13
b11	1	2	2
b12	1/2	1	3
b13	1/2	1/3	1

결함 자료 수집 및 식별 항목간 비교 메트릭스는 b11, b12, b13 열을 기준으로 다른 항목과 중요도를 비교하게 된다. 따라서 b11과 b12는 b11가 2배 많게 분석이 되었다. 그 이유는 b11은 N 이고, b12는 L이므로 두 항목간의 등급차이가 1이므로 중요도 차이 비율은 2배가 된다.

b12와 b11은 등급이 L과 N으로 1/2배 차이가 발생한다. 또한 b13과 b2는 등급이 H와 L로 1/3배 차이가 나타나게 된다.

결함 자료 수집 및 식별을 수행하기 위한 목표 중 어떤 목표가 중요한가를 산출하여야 한다. 따라서 이와 같은 중요한 목표를 산출하기 위하여 기하 평균을 이용하였다. 결함 연관성 분석 항목의 목표 중요도는 다음과 같이 산출되었다.

〈표 16〉 결함 연관성 분석의 하위 목표 중요도 산출표

항 목	산 출 값
b11	$(1 \times 2 \times 2)^{1/3} = 4^{1/3} =$ **1.58**
b12	$(1/2 \times 1 \times 3)^{1/3} = 3/2^{1/3} = 1.14$
b13	$(1/2 \times 1/3 \times 1)^{1/3} = (1/6)^{1/3} = 0.56$

b11(결함 발생 시 명확한 SPEC이 없는 경우 이에 대한 처리) 하위 목표가 결함 연관성 분석에서 가장 중요한 달성 목표로 분석되었다. 따라서 b11(결함 발생 시 명확한

SPEC이 없는 경우 이에 대한 처리)이 결함 연관성 분석 목표를 달성하기 위하여 우선순위가 가장 높고, 달성되어야 할 하위목표로 나타났다. 그리고 b12(결함발생으로 인하여 변경사항 관리 방안)가 1.14, b13(결함의 원인에 대한 책임 전가)이 0.56의 순서로 달성되어야 목표가 분석되었다. 또한 b11(결함 발생 시 명확한 SPEC이 없는 경우 이에 대한 처리), b12(결함발생으로 인하여 변경사항 관리 방안), b13(결함의 원인에 대한 책임 전가)의 목표를 달성하기 위해서는 다음과 같은 해결책을 제시할 수 있다.

〈표 17〉 결함 연관성 분석 목표를 위한 해결책

결 함 내 용	해 결 책
1. 결함 발생 시 명확한 SPEC이 없는 경우 이에 대한 처리	1. 결함 종류 분류 2. 주기적인 결함 조사 3. 결함 관리 정책 수립
2. 결함발생으로 인하여 변경사항 관리 방안	1. 결함의 영향도 분석 2. 변경 이력 관리
3. 결함의 원인에 대한 책임 전가	1. 결함 식별 2. 결함 원인 분석 3. 결함 원인의 트리거 설계 4. 작업의 책임과 권한 명시

Boundary 문제를 해결하기 위한 AHP 알고리즘은 다음과 같다.(결함 연관성 분석을 위한 boundary 문제 해결 절차) 전제 조건으로는 목표가 우선적으로 선정이 되어야 한다. 그리고 목표에 대한 하위 목표를 결정한 후, 비교 행렬에 의하여 중요도를 결정한다. 이와 같은 내용의 알고리즘은 다음과 같은 단계로 나타낼 수 있다.

단계 1 : 문제를 계측적으로 분해-문제의 트리 설계
단계 2 : 비교 행렬 생성
A = (a_{ij} = (w_i/w_j),
I, j는 I번째와 j번째 항목의 중요성 지칭
1 : Equal
3 : Moderate
5 : Strongly
7 : Very Strongly
9 : Extreme
2, 4, 6, 8 : 중간값
역 행렬 계산 : if $a_{ij} = w_i/w_j$ then
$a_{ji}=1/a_{ij} = w_j/w_i$)
단계 3 : 비교 중요성 추정
(1) 행렬 A의 기하 평균 계산
(2) 각 요인에 대한 비교 중요성 계산
단계 4 : 요인별 측정값 계산
단계 5 : 측정값과 비중의 통합 계산

$$V = \sum_{j=1}^{n} w_j a_j,$$

w_j는 비중, a_j는 성취도

4. 사례연구

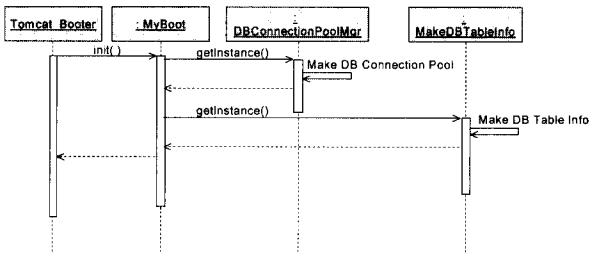
4.1 결함 처리를 위한 시스템 설계

패키지의 수가 실제 사용되는 클래스의 수에 비해서 다소 많은 듯하나 이는 차후 버전을 고려하여 설계하였다. 의 시스템 설계 패키지 설명은 다음 표와 같다.

〈표 18〉 시스템 설계 패키지 설명 테이블

패키지명	설 명
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 다음과 같이 만들어 졌다.

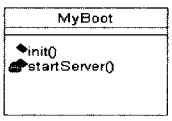


(그림 6) 부팅의 순서

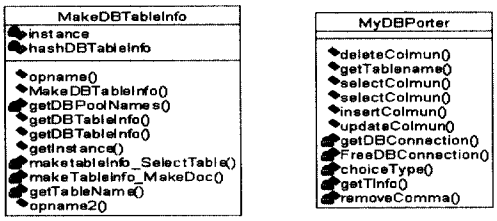
부팅 순서는 접속 풀(Connection Pool)를 생성하고 그 접속 풀(Connection Pool)를 이용해서 테이블의 정보를 취득한다.

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 7) MyBoot의 클래스 다이어그램



(그림 8) MakeDBTableInfo와 MyDBPorter의 클래스 다이어그램

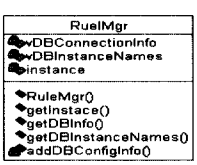
MakeDBTableInfo는 테이블에 대한 정보를 정리하고 이를 돔 트리 객체(DOM Tree Object)를 만들고 이를 저장한다. 그리고 이와같은 객체(Obejct)를 이용해서 데이터베이스 포터(DBPorter)는 데이터베이스와의 트랜잭션(Transaction)을 담당한다. 클래스 다이어그램의 속성과 메소드 설명은 다음 표와 같다.

〈표 19〉 MakeDBTableInfo 클래스 다이어그램의 속성과 메소드 설명 테이블

Attributes명	설 명
instance	MakeDBTableInfo 자체를 나타내는 것
hashDBTableInfo	관련된 테이블에 대한 정보를 저장하고 있는 객체
Method 명	설 명
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 20〉 MyDBPorter 클래스 다이어그램의 메소드 설명 테이블

Method 명	설 명
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

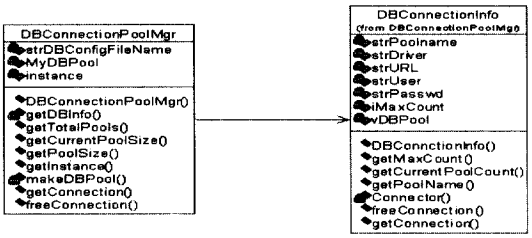


(그림 9) RuelMgr의 클래스 다이어그램

본 구현에서 사용되는 모든 객체(Object)에 대한 기초 자료를 Config 파일에서 읽고 이를 저장한다. (그림 4-10)의 클래스 다이어그램의 속성과 메소드 설명은 다음 표와 같다.

〈표 21〉 RuelMgr 클래스 다이어그램의 속성과 메소드 설명 테이블

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



(그림 10) DBConnectionPoolMgr과 DBConnectionInfo 클래스의 다이어그램

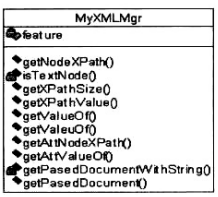
본 구현에서 사용하는 ConnectionPool는 일반적으로 상업용 DBMS에서 제공하는 것과는 틀리다. 일반적으로 가장 많이 사용하는 Hashtable이란 Java의 Object를 사용하였고, 아울러 제공된 Connection을 회수하고 그 Connection을 다시 사용한다. 즉, Config에 Connection의 수 만큼만 만들고 더 이상은 만들지 않는다. (그림 10)의 클래스 다이어그램의 속성과 메소드 설명은 다음 표와 같다.

〈표 22〉 DBConnectionPoolMgr 클래스의 다이어그램의 속성과 메소드 설명 테이블

Attribute	설 명
MyDBPool	ConnectionPool를 담고 있는 객체
instace	
Method명	설 명
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 23〉 DBConnectionInfo 클래스의 다이어그램의 속성과 메소드 설명 테이블

Attribute	설 명
strPoolname	풀(Pool)의 이름
strDriver	JDBC Connection을 위해서 필요한 정보
strURL	
strUser	
strPasswd	
iMaxCount	풀(Pool)에 저장될 Connection 수
vDBPool	DBConnectionPool
Method명	설 명
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 11) 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

결함 관리를 위한 OT 프레임워크의 데이터베이스 내용은 다음 표와 같다.

〈표 24〉 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 25〉 데이터베이스 필드 테이블-2

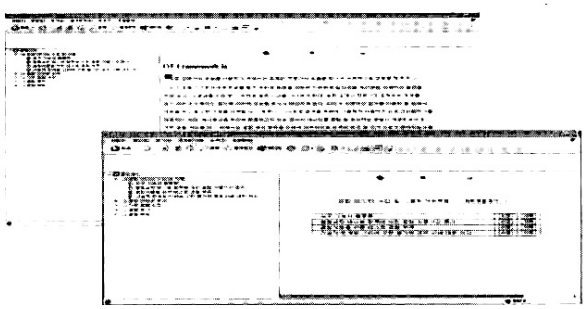
Field Name	Type	Comment
SeqID	int(10)	각 필드의 Key
ParentSeqID	int(10)	수정되었을시에 원본의 SeqID
MenuID	varchar(2)	main_menu의 MainID
ISSubNode	varchar(1)	하위 노드의 존재 유무 Y/N
SubNodeID	int(3)	하위 노드로 존재할시에 부여되는 ID
SubMenuID	varchar(4)	하위 노드로 존재할 시에 상위 노드에 해당되는 ID
SubName	varchar(100)	현 필드에 해당되는 이름
F1	varchar(100)	Parent 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 26〉 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)

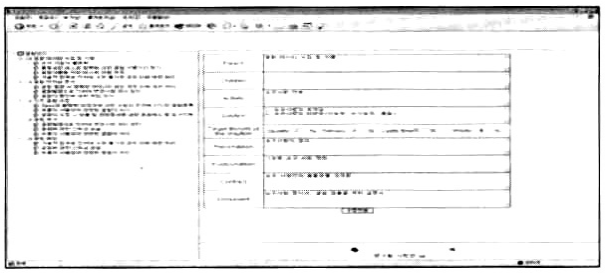
4.2 화면 설계

로근인 후 OT 프레임 워크의 개요와 하위 항목을 추가, 수정, 삭제할 수 있는 화면은 다음과 같다.

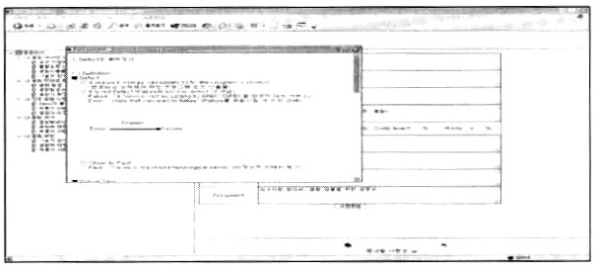


(그림 12) 항목 추가 기능 및 내용 뷰어(Contents Viewer)

해당되는 결함 항목에 대한 상세 정보와 용어정리의 화면은 다음과 같다.

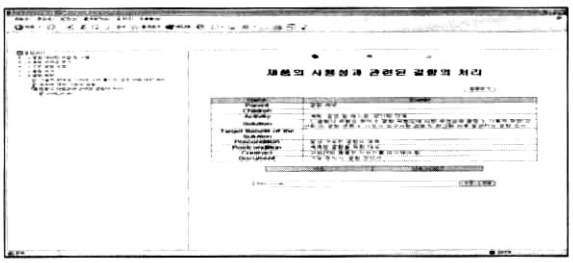


(그림 13) 결함 항목 등록 화면

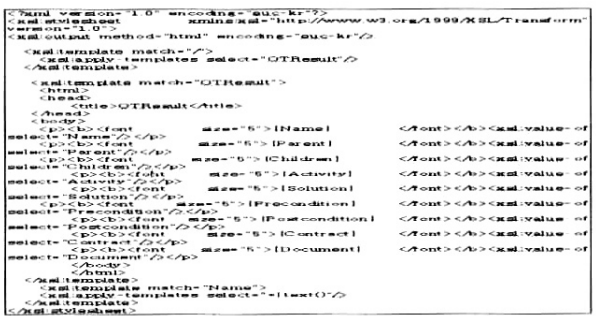


(그림 14) 용어 정리

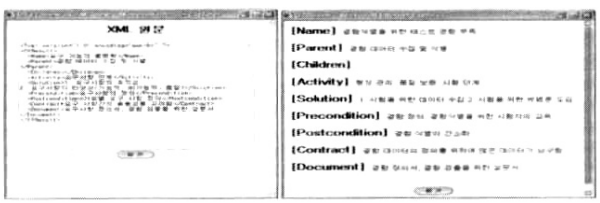
하위 노드의 추가를 위한 정보 제공과 보고를 위한 XML의 제공 화면은 다음과 같다. 수정과 삭제를 통하여 하위 항목을 제어할 수 있게 된다.



(그림 15) 하위 노드 추가



(그림 16) 보고서 생성을 위한 XML 소스 코드



(그림 17) 웹 브라우저에서의 XML 형태

5. 평가 및 결론

기존에 존재하는 결함 관리 방법은 결함 데이터를 수집한 후에 경향을 분석하는 단계의 것이다. 따라서 결함 자료를 활용하여 결함을 제거하고 예방하는 단계의 분석 작업은 수행하지 않고 있다. 기존 방법과 제시된 방법은 다음과 같이 비교 될 수 있다.

〈표 27〉 기존방법과 결함 관리 OT 프레임워크의 비교

	기존 방법	결함 관리를 위한 OT프레임워크
결함 자료의 활용 범위	단순한 결함 개수 산정	산정된 결함으로 경향 분석 수행
자료의 내용	실제 자료만 가짐	실제 내용을 분석한 형태
사용자의 의견 수렴	의견 수렴이 불가능함	의견을 반영할 수 있음
자료의 상세화	수치자료등으로 상세화 됨	수치자료가 부족함
보고서 형식	2차원 수준	2차원 수준 및 재사용을 위한 형식(XML) 마련

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

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

첫째, 제품의 품질 보증을 통한 고객 만족과 제품의 유연성 제고 및 생산성을 향상시킬 수 있다.

둘째, 품질과 인도시간을 조정하여 시장 확장과 재무성과를 최대화 시킬 수 있다.

향후 연구로는 사례연구의 자료를 30개 이상 수집하여 통계적인 분석을 F 분포에 의해서 수행하는 것이다. 이때 설문서 조사 및 분석에 의한 지연요소는 Agile 방법론을 적용하여 간소화 시키는 방안을 연구하겠다. 그리고 결함 처리 이외의 목표를 설정하여 OT 프레임워크를 설계하고, 결과적으로 소프트웨어 프로세스 개선이라는 OT 프레임워크를 설계하는 것이다. 또한 비용 산정과 결함 처리 도구를 OT 프레임워크와 연동하는 시스템을 구축할 계획이다. 그리고 이와 같은 도구를 서비스 해줄 수 있는 3차원적인 그래픽 도구를 활용할 수 있도록 연구할 계획이다.

참 고 문 헌

[1] 이경환, HDC를 위한 모델링, 제5회 한국정보과학회 소프트웨어공학 연구회, Feb., 2003.

[2] Annual Research Review, USC/CSE workshop reports, Oct., 2002.

[3] Software Process Improvement Forum, KASPA SPI-7, dec. 2002.

[4] George Eckes, The Six Sigma Revolution, John Willey & Sons, 2001.

[5] SPICE Assessment in korea, The korea SPICE, May, 2002.

[6] 신경애, "결함중요도 단계를 고려한 소프트웨어 신뢰도 성장 모델에 관한 연구", 컴퓨터 산업교육기술학회 논문지, Vol.3, No.7, pp.0837-0844, Jul., 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, Apr., 2002.

[9] N. Fenton and N. Ohlsson, "Quantitative analysis of faults and failures in a complex software system," IEEE Trans. Software Eng., 26, pp.797-814, 2000.

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

[11] Vouk, Mladen, A., "Software Reliability Engineering," Tutorial Notes? Topics in Reliability & Maintainability & Statistics, 2000 Annual Reliability and Maintainability Symposium, Los Angeles, CA, January, 2000.

[12] Padberg, "A Fast Algorithm to Component Maximum Likelihood Estimates for the Hypergeometric Software Reliability Model," Asia-Pacific Conference on Quality Software APAQS 2, pp.40-49, 2001.

[13] Vamder Wiel, Votta, "Assessing Software Designs Using Capture-Recapture Methods," IEEE Transaction on Software Engineering, 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, Nov., 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 L. SAATY, "Analytical Planning," RWS PUBLICATIONS, 1985.

[21] V. Basili, G. Caldiera and D. Rombach, "The Experience Factory," Encyclopedia of Software Engineering, Wiley 1994.

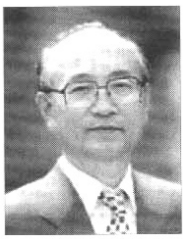
[22] V. Basili, G. Caldiera and D. Rombach, "The Goal Question Metric Approach," Encyclopedia of Software Engineering, Wiley, 1994.

[23] Victor R. Basili, "The Experience Factory and its Relationship to Other Improvement Paradigms," Lecture Notes in Computer Science 717, Software Engineering ESEC'93, 4th European Software Engineering Conference Garmish-Partenkirchen, Germany, September, 1993, December, 1991.

이 은 서

e-mail : esleee@object.cau.ac.kr
1999년 중앙대학교 컴퓨터공학과(석사)
2001년~현재 중앙대학교 컴퓨터공학과 (박사과정)
관심분야 : CBD, Formal method, Quality model, SPI(Defect Analysis)

이 경 환



e-mail : kwlee@object.cau.ac.kr
1964년 중앙대학교 이과대학 수학과(학사)
1966년 중앙대학교 이과대학 수학과(석사)
1980년 중앙대학교 대학원 수학과(박사)
1982년~1983년 미국 AUBURN 대학교 객원교수
1986년~1986년 독일 BONN 대학교 객원교수
1992년~JCSE '92 Conference Chairman 서울
1993년~1995년 중앙대학교 공과대학 학장
1994년~1995년 중앙대학교 정보산업대학원 원장
1998년~2000년 중앙대학교 정보통신연구소 소장
1999년~2000년 한국 정보과학회 회장
2000년~2001년 중앙대학교 총무처장
2001년~2002년 미국 USC 대학교 객원교수
2003년 1st ACIS 국제 학술 대회 Program cochair
2003년 SERA 2003 Program chair
2004년 11th Asia-Pacific SE Conference chair
2004년 11th ASPEC 2004 General chair
2004년 공동 편집자, Springer LNCS 3026, Software Eng and Appl.
1992년~현재 한국표준 소프트웨어 공학(SC7) 위원
1992년~현재 ISO/SC7/WG10 SPICE 한국 위원장
관심분야 : CBD Architecture, Software Process