Bug Tracking 시스템을 활용한 테스트 및 결함관리

안유환*, 김신애*, 허희정*
*(주)핸디소프트
e-mail: ywahn@handysoft.co.kr

Mnanaging testing function and defects using a Bug Tracking System

Yuwhoan Ahn*, Shine Kim*, Heejung Huh*
*HandySoft Corporation

요 약

소프트웨어의 품질을 향상시키기 위해 결함을 찾고 시정하는 것을 통제하는데 가장 중요한 요소가 바로 오류/장애/실패(통칭하여 결함(defects)를 효과적으로 추적하는 것이며, 결함을 추적하기 위해서는 효과적인 결함 추적시스템(Defect Tracking System)의 도입이 필요하다. 기존의 결함 추적 시스템은 고객/사용자로부터의 문제보고, 테스팅 기간 중의 결함보고, 변경관리 기능 등을 통합적으로 제공하고 있지 못하다는 점과 테스팅을 포함한 개발과정의 정량적 통제를 위한 데이터를 효과적으로 축적.제공하지 못하고, 더욱이 결함의 근본적인 원인을 찾아 해결할 수 있도록 하기 위한 정보를 효과적으로 제공하지 못하고 있는 문제가 있다. 본 논문에서는 이러한 기존의 결함 추적 시스템의 문제점을 해결하고자 설계/구축되어 현재 사내에서 활용되고 있는 결함관리 시스템인 Promise System을 소개하고자 한다. Promise 시스템은 고객의 불만 및 요구사항, 인스펙션 및 테스팅 시의 결함 및 요구사항에 대하여 각 담당자별로 업무를 할당, 추적할 수 있게 하고, 결함 및 요구사항에 관한 각종 상태정보와 통계정보를 제공하여, 각 개발 및 테스팅 단계별로 결함 제거 목표의 수립, 정량적 통제, 결함 원인 분석을 통한 프로세스 개선 등을 지원하는 시스템이다.

1. 서론

소프트웨어의 품질을 향상시키기 위해 결함을 찾고 시정하는 것은 분명히 소프트웨어 개발 및 유지보수에서 많은 비용을 차지하고 있다. 이러한 통상적인 개발 과정에서의 비용의 반 이상을 차지하고 있다. 개발과정에서 50%의 결함 감소를 보이도록 결함 예방조치를 시행한 어떤 프로젝트에서는 릴리즈 이후에 78%의 결함 감소효과를 보이는 사례도 있다. 이는 개발과정에서 줄인 결함에 더하여 테스트 과정에서는 이미 대부분의 사소한 결함은 이미 제거되어 보다 더 집중하여 주요 결함들을 검증.제거할 수 있기 때문이다[7].

테스팅은 결함 발견, 측정(measurement) 정보의 제공 등의 이유로 매우 중요한 지원 기능으로 인식되고 있으며, 테스팅을 효과적으로 통제하기 위해서는 테스팅이 적절히 이루어지고 있는지를 확신할 수 있도록 필요한 피드백을 할 수 있는 정보를 제공해야 한다[2]. 테스팅의 효과성(effectiveness)을 측정하기 위한 주요 수단은 오류(error)/장애(fault)/실패(failure)의 추적, 경향(trends)의 분석, 테스팅 비용의 추적, 테스팅 상태의 추적, 테스트 문서화 등이 있으며, 이중에서도 테스팅 기능을 통제하는데 가장 중요한 요소가 바로 오류/장애/실패(통칭하여 결함(defects)를 효과적으로 추적하는 것이다[2][8].

결함을 추적하기 위해서는 효과적인 결함 추적시

스템(Defect Tracking System)의 도입이 필요하다. 근래까지도 결함 추적 시스템은 대규모의 소프트웨어 개발에만 사용되는 것으로 인식되고 있었다. 소규모 프로젝트에서는 메일이나 게시판을 사용해서도 결함을 추적할 수 있었지만, 수많은 결함들이 어떻게 처리되었고 어떤 이유로 인하여 기각이 되었는지. 문제점은 무엇인지. 누가 담당을 하였는지, 처리되는데 얼마나 시간이 소요되었는지에 대한 정보까지 다 기억할 수는 없기 마련이다. 특히 메일의 경우에는 안 읽으면 또 그만이어서 결함이 묻혀서 사라질 수도 있는 것이다. 때문에 최근 많은 회사들은 시간을 절약하고, 생산성을 향상하면서, 고객의 만족을 높이기 위하여 통합된 결함 추적 시스템을 찾아 적용하고 있다.

그러나 기존의 결함 추적 시스템은 Knowledge Base, Trouble Ticketing, Bug Tracking System, Workflow Management 등의 여러 형태로 개발되어 활용되고 있지만[6][7], 고객/사용자로부터의 문제보고, 테스팅 기간 중의 결함보고, 변경관리 기능 등을 통합적으로 제공하고 있지 못하다는 것이다. 이러한 것은 "고객/사용자와 개발자간의 인터페이스를 담당하는 관리자는 이러한 결함관리 시스템과 변경관리 프로세스가 없으면 그들 시간의 80% 이상을 데이터를 모으는 데에만 소비한다[7]"는 점을 고려하면 정말 문제가 되는 부분이다.

또한 기존의 결함추적 시스템들은 테스팅을 포함한 개발과정의 정량적 통제를 위한 데이터를 효과적으로 축적.제공하지 못하고, 더욱이 결함의 근본적인 원인을 찾아 해결할 수 있도록 하기 위한 정보를 효과적으로 제공하지 못하고 있다.

본 논문에서는 이러한 기존의 결함 추적 시스템의 문제점을 해결하고자 설계/구축되어 현재 사내에서 활용되고 있는 결함관리 시스템인 Promise System을 소개하고자 한다. Promise 시스템은 고객의 불만 및 요구사항, 인스펙션 및 테스팅 시의 결함 및 요구사항에 대하여 각 담당자별로 업무를 할당, 추적할 수 있게 하고, 결함 및 요구사항에 관한 각종 상태정보와 통계정보를 제공하여, 각 개발 및 테스팅 단계별로 결함 제거 목표의 수립, 정량적 통제, 결함 원인 분석을 통한 프로세스 개선 등을 지원하는 시스템이다.

제 2장에서는 기존 시스템의 문제점을 분석하고 결함관리 시스템의 요구사항을 제시하며, 제 3장에서는 설계/구축된 Promise 시스템의 기능과 제공되는 정보를 설명한다. 마지막으로 제4장에서는 시스템의 활용 현황과 현 시스템의 향후 개선 방향등을 제시하고자 한다.

2. 결함 관리 시스템 요구사항

2.1 버그 추적 시스템

ANSI/IEEE 표준에서는 테스팅 기간 중에 발견되는 이상을 기록하는 2개의 문서: 테스트 로그(Test Log)와 테스트 사건 보고서(Test Incident Report)를 명시하고 있다[5]. 대부분의 조직에서는 이와 유사한 형태의 여러 결함 경고, 문제보고 또는 추적 양식이 있지만, 핵심은 테스팅 기간 중 또는 시스템 설치(또는 릴리즈) 이후에 결함정보를 지속적으로 정형적으로 추적하는 수단을 가지는 것이다.

기존의 결함추적 시스템을 보면, Knowledge Base, Trouble Ticketing, Bug Tracking System, Workflow Management 등의 여러 형태가 있어 다른 목적으로 활용되고 있지만, 기본적으로는 문제 및 버그를 해결하기 위한 작업을 추적하고자 하는 것이다[3]. 본 논문에서는 관련이 많은 Trouble Ticketing, Help-Desk Management/Call Tracking ,Bug Tracking , workflow management만 간단히 아래에 설명한다.

- Trouble Ticketing

  Trouble-ticketing system은 문제(결함을 포함)들에 대한 독립적인 작업 항목(work item)을 추적하는 간단한 기능으로 국한되어 있으며, 작업항목을 다른 사람들에게 전달하기도 한다. 작업들이 독립적으로 간주되기 때문에 open, in-progress, closed 등의 간단한 상태정보만 관리한다.

- Help-Desk Management/Call Tracking

  대부분의 Help-Desk Management system은 다양한 고객 추적 기능을 부가한 것 외에는 Trouble-Ticketing system과 비슷하다. 주로 자세한 고객정보에 대한 기록, 문제에 대한 검색 능력, FAQ 참조 기능, 시간 추적 등의 기능을 제공하고, 고급 제품이 경우에는 일부 workflow 기능(업무 전달이 자동화 기능)과 상위 관리자로 보고하는 기능을 가지고 있다.

- Bug Tracking

  버그 추적 시스템(Bug-Tracking System)은 독립적인 작업을 추적한다는 점에서는 Trouble-ticketing system과 비슷하나, 보통 더 많은 책임과 역할을 정의하고 있고(예를 들면, 프로그래머, 통합자/빌더, 시험자, 관리자 등), 각 담당자의 역할을 제한하고 있다. 일부 시스템들

은 버전관리 또는 현상관리 시스템과 통합되어 있다.

- Workflow Management

Workflow Management system은 여러 가지 역할과 권한을 정의할 수 있다는 점에서는 Bug-Tracking system과 비슷하지만, 문서관리 시스템과 강하게 결합되어 있어 버전이 관리되는 문서들의 생성, 수정, 전달을 지원하며, 각 작업의 비용 산정, 측정을 지원한다.

일반적으로 버그 추적 시스템은 다음과 같은 기능을 가지고 있으며, 버그 추적시스템은 Bugzilla, Tracker, Debian Bug Tracking System, Open Track, Time Sheets for Networks (TSN), journyx Time 등의 수많은 공개/free 소프트웨어와 TestTrack Pro, Rubicon Tracker, Keystone 등의 여러 상용 소프트웨어가 있다[6][7][10].

- 문제의 발견 및 해결과정까지의 추적
- 각 문제의 관련 파일의 첨부
- 문제의 분석 보고
- 텍스트 기반 문제의 검색
- 버그의 상태 변경의 알림(메일링)

2.2 결함관리 시스템의 요구사항

기존의 버그 추적시스템은 그러나 몇가지의 문제점을 가지고 있다.

첫째로, 문제의 보고는 고객/사용자들로부터 접수될 수도 있고, 개발과정에서 인스펙션 시에 테스팅 중에 보고될 수도 있다. 또한 보고된 내용 중에는 해결과정에서 결함의 시정조치로 연결되거나 또는 요구사항으로 또는 프로세스의 개선사항으로 연결되기도 한다. 따라서, 고객의 불만/문제를 추적하기 위한 problem tracking 시스템, 제품/시스템의 요구사항을 관리하기 위한 요구사항 관리 시스템, 개발과정에서의 결함 수정을 위한 결함 추적 시스템, 인스펙션 관리 시스템 등이 통합되어 제공되어야 한다. 이는 CMMI에서 예로 제시하는 관련되는 데이터를 시정조치를 요구 내용(프로젝트 관리 상의 문제), 고객/사용자에게서 보고되는 결함, 동료검토에서 발견된 결함, 테스팅에서 발견된 결함, 프로세스 능력 상의 문제 등으로 제시하는 것을 보아서도 알 수 있다[3].

둘째로, 대부분의 시스템들이 작업 항목에 대한 추적과 각 작업자에게의 업무 전달을 지원하고 있지만, 개발과정에서의 각 개발자의 역할(개발자, 프로덕트 관리자, 빌더, 테스터, 품질보증 담당자, 프로젝트 리더, 프로젝트 관리자) 별로 필요한 검토/승인 과정을 모두 지원하지 못하고 있다. 이는 CMM에서 요구하는 요구사항의 검토, 문제 및 해결과정의 기록, 시정조치의 추적 및 기록 등의 요구사항을 완벽하게 지원하지 못한다.

셋째로, 개발과정에서의 결함을 줄이기 위한 인스펙션 활동, 테스팅 활동 등을 정량적으로 관리하기 위한 데이터의 수집. 분석 기능이 부족하고, 결함의 근본적인 원인을 파악할 수 있는 데이터의 수집 및 분석, 제공 기능이 부족하다.

2.3 데이터의 수집 및 분석 요구사항

테스팅 시 또는 릴리즈 이후의 결함은 기본적인 아래와 같은 몇가지 질문에 응답할 수 있도록 분석되어야 하고 추적과 피드백 기능을 지원할 수 있어야 한다[1].

- 보고된 내용이 결함이면
  - 어떻게 발견되었는지?
  - 무엇이 잘못된 것인지?
  - 누가 결함을 만든 것인지?
  - 언제 만들어진 것인지?
  - 왜 조기에 발견되지 않았는지?
  - 어떻게 하면 사전에 예방할 수 있었는지?
- 보고된 내용이 결함이 아니면
  - 왜 그러한 내용이 보고되었는지?
  - 어떻게 하면 사전에 예방할 수 있었는지?

Gaffney는 38개의 관리척도를 총 50개의 CMM 질문항목에 대하여 SW-CMM 각 KPA별로 활용가능한 척도를 연결하였다. CMM의 성숙단계별로 품질 및 생산성의 측면에서 관리되어야 하는 척도를 정리하면 다음의 <표 1>과 같다.

프로젝트에서의 결함예방 활동에 있어서, 각 개발단계에서 발견된 결함의 분석은 이후의 개발단계에서의 결함발견과 많은 관계가 있다. 최소한 결함의 심각도와 유형에 대한 결함 분석은 이루어져야 한다[11]. Infosys에서는 결함의 분포(개발 단계별 결함 수 및 전체 결함 대비 비율을 산정치 및 실제치로 관리), 결함제거 효과성(결함 유입/발견 매트릭스 포함), 결함 심각도별 분포, 결함 유형별 분포 등을 관리하고 있다. 이때 결함의 유형은 decision issue, hard-code, Initialize, Issue, logic, Standards, User Interface, Other 등으로 구분하고 있다. 이러한 분석의 결과로 어떤 결함유형이 너무 많으면 그러한 유형의 결함을 예방할 수 있도록 프로세스가

개선되어 질 수 있다.

<표 1> CMM 단계별 관리 척도.

KPA	번호	관리척도	범 주
CM	1	ECP의 수	안정성
	2	(정의된 요구사항 수)/(전체 요구사항 수)100	
PTO	1	결함 또는 에러/KSLOC(실제 또는 예측된 KSLOC) [코딩 단계]	품질
	2	예상 결함/KSLOC [배포 단계]	
	3	(처리된 PTR/전체 PTR)100	
	4	PTR/KSLOC [통합시험 단계]	
	5	PTR/KSLOC [시스템시험 단계]	
PR	1	결함 또는 에러/KSLOC(실제 또는 예측된 KSLOC) [PDR, DDR단계]	품질
	2	(완결된 SAI/전체 SAI)100 [코딩단계에서 발생된 SAI 추적]	
	3	결함이나 에러/KSLOC(실제 또는 예측된 KSLOC) [코딩 단계]	
QPM	1	계획 대비 실제 결함 또는 에러수 (결함 또는 에러/KSLOC) [PDR, DDR단계]	품질
	2	계획 대비 실제 결함 또는 에러수 (결함 또는 에러/KSLOC) [코딩 단계]	
	3	공정 데이터베이스 설립(전체 프로젝트 공정척도 대상)	경험 데이터 베이스
	4	제품에 잔존하고 있는 에러의 분포와 특성을 파악(코딩과 시험 단계에서의 에러 데이터를 토대)	
	5	주요 공정단계를 위한 소프트웨어 생산성 분석	
SQM	1	결함 또는 에러/KSLOC (실제 또는 예측된 KSLOC) [PDR, DDR]	품질
	3	검토 데이터(review data) 분석 [PDR, DDR, 코딩 단계]	

<용어설명>
DDR: Detailed Design Review, ECP: Engineering Change Proposal, FP: Function Point, KSLOC: Thousand Source Lines of Code, LH: Labor Hour, LM: Labor Month, MIPS: Million Instructions Per Second, PTR: Program Trouble Report, PDR: Preliminary Design Review, SAI: Software Action Item

Humphrey는 결함이 발견되었을 때와 해결이 종료되었을 때 다음과 같은 정보를 관리할 것을 제안하고 있다[12];

- 결함 발견 시: 결함 id, 결함이 발견된 제품/릴리즈/Driver id.(릴리즈의 subset)/Function, 결함 발견 단계, 결함 유입 단계, 품질개선팀 id., 문제보고 번호, 발생 일자, 분석자, 분석 유형, 결함 해결 상태
- 결함 종료시: 종료 일자, 원인의 부류, 결함의 기술(description), 결함 원인 기술, 관련 시정조치 번호, 문제 기술(문제 및 시정조치 설명), 관련 시정조치 활동 기록

특히, 결함의 조기발견을 위한 개발 및 테스트 프로세스의 통제, 결함 예방을 위한 결함의 근본적 원인 분석 등은 매우 중요하다. 이는 소프트웨어 개발 능력과 품질을 지속적으로 개선하기 위한 CMM의 high maturity인 level 4와 level 5의 요구사항에서도 알 수 있다. CMM level 4와 level 5를 달성하는데 기본적인 것들이 CMMI의 Staged representation을 참고하면, 정량적 프로젝트 관리(Quantitative Project Management) 프로세스 영역(Process Area)와 원인분석 및 해결(Causal Analysis and Resolution) 프로세스 영역이다[3]. 이는 각각 SW-CMM에서는 소프트웨어 품질관리(SQA) KPA(Key Process Area), 결함 예방(DP: Defect Prevention) KPA에 해당된다.

결함과 문제의 분석을 위하여는 파레토 분석, 히스토그램, 인과관계 다이어그램(fish bone diagram), check sheet 등이 있으며, 기본적으로 결함 데이터가 기록되어야 하는데 이때 필요한 사항들은 다음과 같다[9]: 결함의 기술(description), 결함 원인의 기술(description), 결함 원인 부류, 결함 유입 단계, 결함 발견 단계, 문제 해결책의 원천과 부류 및 그 내용

마지막으로, S/W 개발공정에서의 효과적인 결함 제거는 S/W 개발 프로젝트 성공의 중요 요인이다. 이는 개발공정의 초기단계에서의 결함제거가 중요하기 때문이기도 하다. 이를 위해서는 각 개발단계별 결함제거효과성을 측정하기 위한 아래와 같은 수단이 필요하다;

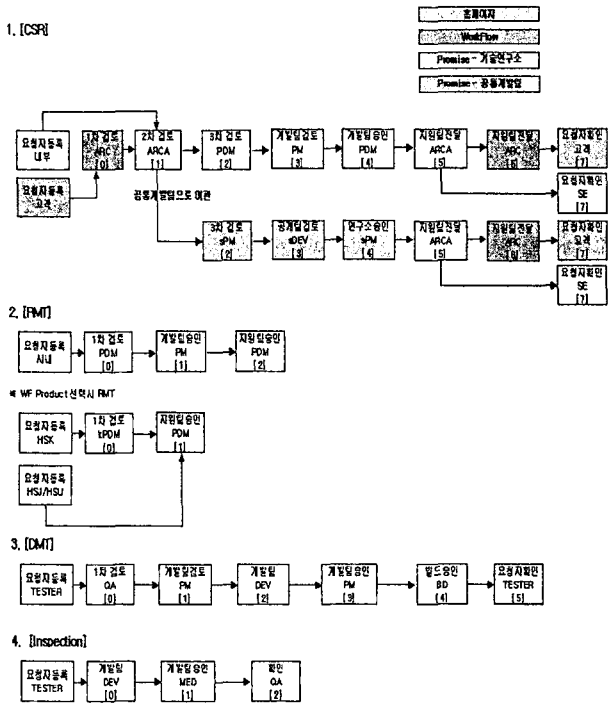
- 결함의 원천(origin)과 개발단계(where found)를 교차시킨 행렬 형태의 테이블(Origin/Where found Matrix)의 사용
- 단계별 결함제거모형(phased-based Defect Removal Model: DRM)
- 결함투입(defect injection), 결함제거(defect removal), 결함제거효과성(effectiveness) 등 세개의 척도 사이의 상호관계를 요약하는 행렬 형태의 테이블을 단계적으로 작성
- 유사한 개발환경을 가진 과거의 개발경험 기초

3. 요구사항 및 결함 관리 시스템: Promise System

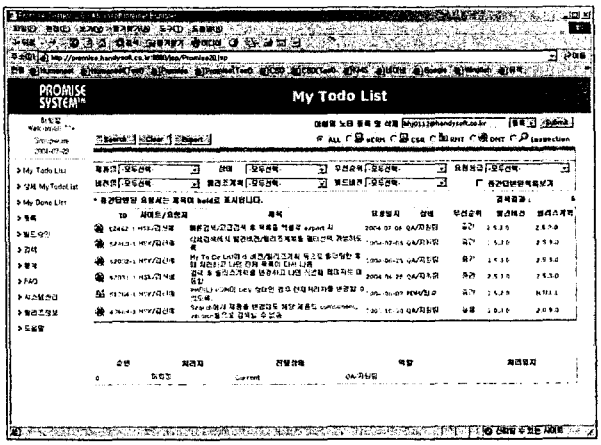
3.1 Promise 시스템의 주요 기능

Promise 시스템은 요구사항 및 결함 관리를 위하여 개발된 시스템으로 사내 연구본부와 개발본부가 사용하는 도구이다. [그림 1]과 [그림 2]는 Promise 시스템의 주요 프로세스와 main menu 화

면을 보여주고 있다.



[그림 1] Promise 시스템의 주요 프로세스



[그림 2] Promise 시스템 main 화면

Promise로 처리되는 요청서의 경로는 eCRM, CSD, Promise 자체 등록 이렇게 3가지의 Process가 있어, 고객 요청관리 시스템, 통합고객관리 시스템으로부터 개발이 필요한 요구사항 및 결함보고를 넘겨받아 처리하고 그 결과를 해당 시스템으로 돌려주는 역할을 한다. 또한 Promise 시스템의 내부적으로는 아래의 주요 서브시스템으로 구분되어 요구사항관리와 결함관리(인스펙션으로부터의 결함보고 포함) 기능을 제공한다.

- CSR : 릴리즈 이후에 발생하는 사이트 요구사항과 결함 등을 처리
- RMT : 개발 단계의 제품에 대한 요구사항을 관리
- DMT : 개발 단계의 제품에 대한 결함을 등록하여 처리
- Inspection : 개발 단계의 제품에 대한 동료 검토 결과를 등록하여 관리

참고로, eCRM은 제품과 관련된 고객의 문의 및 요청사항을 홈페이지 고객의 소리를 통해 접수 받아 정의된 내부 프로세스에 따라 업무를 처리하는 시스템으로, BPM 제품(Handy BPM)으로 구현되어 있다. 접수된 고객의 요청서는 상황에 따라 Promise (요구사항 및 결함 관리) 시스템으로 이관되어 개발팀에서 처리를 하게 된다. CSD는 통합 고객 관리 시스템으로 제품에 대한 문의지원 및 제품 설치 정보 등 고객과 관련된 정보를 통합 관리하는 도구이며, 마찬가지로 개발과 관련된 기술 문의 및 요구사항인 경우에는 Promise(요구사항 및 결함 관리) 시스템으로 이관되어 개발팀에서 2차 처리를 하게 된다.

3.2 Promise 시스템의 상세 기능 및 추적 관리

Promise 시스템은 다음과 같은 주요 기능들을 가지고 있다[1]:

1. 사용자 로그인: 사내 그룹웨어를 통하여 접속 가능
2. MyTodoList: MyTodoList는 현재 자신이 처리해야 하는 일들을 보여주며(요청서ID, 사이트/요청자, 제목, 요청일자, 상태, 우선순위, 발견버전, 릴리즈계획 등의 항목) 검색과 이메일 노티의 설정 등이 가능함. 목록에서 제목을 클릭하면 자신의 역할에 따라 요청 처리화면이 나타남.
3. 상세 MyTodoList: 현재 자신이 처리해야 하는 일들을 상세하게 보여줌(요청내용과 답변내용, 히스토리까지 상세한 정보).
4. MyDoneList: 자신이 등록했거나 처리한 모든 사항에 대한 요청서 목록을 보여주며, 목록 중에 현재 처리되고 있는 진행 상태를 확인하고 싶을 경우 해당 요청의 [상태]를 클릭하면 현재 진행 중인 상태를 확인 가능.
5. 등록 및 처리 : CSR, RMT, DMT, Inspection 요청서를 등록하고 처리하는 기능으로 각종 해당 담당자가 각종 상태정보와 처리 내역을 역할별로 입력.처리함: CSR 등록, CSR 처리, RMT 등록, RMT 처리, DMT 등록, DMT 처리, Inspection 등록, Inspection 처리
6. 빌드 승인: 각 제품의 릴리즈별 빌더의 역할로 지정된 사람이 빌드 상태의 요청서를 한꺼번에 빌드 승인하는 기능이며, 모든 사용자가 [빌드승인] 메뉴를 통해 현재 빌드 승인 상태의

요청서 목록과 내용을 확인할 수 있으며 빌더만이 [approve] 기능을 수행할 수 있음

7. 검 색: 빠른검색, 상세보고서, 고급검색의 세가지 주요 기능으로 강력한 검색 기능 제공

8. 통 계 : 다양한 통계보고 기능을 제공합니다. 등록현황, 진행현황, 주요통계를 제공함

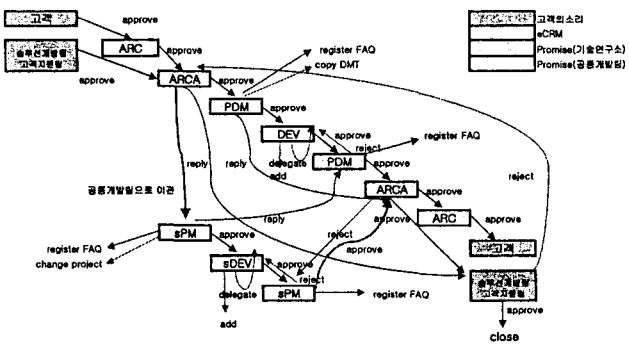
9. FAQ : 요청서의 내용과 답변을 FAQ 게시판으로 등록하여 자주 문의되는 오류 및 요청내용과 그에 대한 답변을 공유 가능함. 요청서 처리화면에서 FAQ 등록 버튼을 클릭하시면 요청내용과 최종답변 내용을 default로 제공하며 FAQ 게시판으로 등록할 내용의 수정 가능

10. 시스템 관리: 사용되는 제품(Product)과 각각의 버전(Version), 기능(Component), 사용자(User) 그룹, 시스템 에서 사용하는 코드(Code), 제품이 설치된 사이트(Site) 등의 정보를 등록하고 관리하는 역할

11. 기타 릴리즈 정보 제공 및 도움말

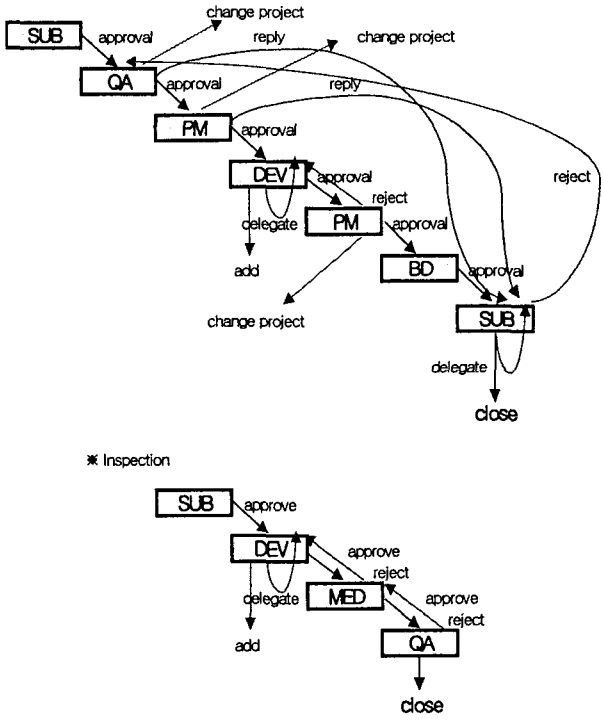
Promise 시스템은 요청이나 결함의 등록 및 처리에 있어서, 각 담당자의 역할(개발자, 프로덕트 관리자, 빌더, 테스터, 품질보증 담당자, 프로젝트 리더, 프로젝트 관리자)별로 필요한 검토/승인 과정을 모두 지원하고 있으며(각 담당자별 엄격한 업무처리 흐름 제어 및 데이터 수정 권한을 정의함), 3.3절에서 설명하고자 하는 상태정보와 통계정보의 제공에 필요한 각종 상태정보를 각 담당자의 역할별로 효과적으로 입력.관리할 수 있게 한다.

[그림 3]은 제품 릴리즈 이후에 발생하는 사이트 요구사항과 결함을 접수 받아 처리하는 CSR 프로세스인데, 요청받은 요구사항과 결함은 그림과 같은 업무처리절차를 통해 다시 요청자에게 처리 결과를 전달한다.

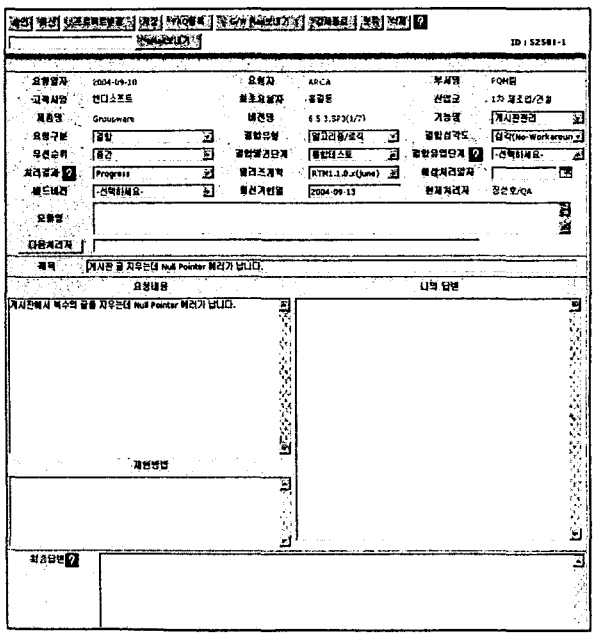


[그림 3] CSR 처리 흐름 및 역할

[그림 4]는 개발 단계의 제품에 대한 결함을 등록하여 관리하는 DMT(Defect Management Tool) 프로세스를 보여주고 있는데, 제품의 릴리즈별 빌더의 역할로 지정된 사람이 빌드 상태의 요청서를 빌드 승인하여 제품에 반영할 수 있다. [그림 5]는 DMT 등록시와 처리시에 입력해야 할 정보를 보여주고 있다.



[그림 4] DMT 처리 흐름 및 역할



[그림 5] DMT 등록 및 처리 입력 사항

DMT 등록 시 입력되는 항목과 주요 항목별 상태 코드는 다음과 같다:

(1) [최초요청자], [회신기한일]을 입력

(2) [제품명], [버전명], [기능명]을 선택

(3) [요청구분]을 선택. Default 값은 "개선"임

구 분	내 용
결 함	오류나 결함
개 선	새로운 기능에 대한 요구사항
지 원	제품에 대한 모든 지원
취 소	사용자의 잘못이나 이해 부족으로 인한 실수
질 문	제품 또는 기능에 대한 질문
중 복	중복된 요청서(Original report 번호 명시)

(4) [결함 유형]을 선택

구 분	내 용
요구사항	누락된 요구사항, 요구사항의 중복/모순, 구현불가능한 요구사항
기 능	구현된 기능이 요구사항대로 동작하지 않거나 명세서 대비 누락/중복된 경우
알고리즘/로직	Internal algorithm이나 logic의 오류
데이터정의/질의	Data Structure의 부적절한 분할, 중복, 명세에 대한 결함, 부적절한 data 선언과 정의, 초기화, 화면과 PGM 간의Data 불일치
인터페이스	모듈 내부의 인터페이스가 정의되지 않았거나 외부 시스템과 연동 시 인터페이스가 정의되지 않았거나 부정확한 데이터 전달, 모듈 간의 부정확한 파라미터 사용, 입력 반영 오류 등
사용자인터페이스	직관적이지 않는 UI, 표준을 따르지 않는 layout과 navigation flow, 메시지 오류 등
성 능	요구되는 실행 효율성을 만족하지 못할 가능성이 있는 결함 (실행속도 저하, 메모리 과다 사용)
기 타	위에 해당되지 않는 경우와 개발 프로세스에 관한 문제 등

(5) [결함발견 단계]을 선택

(6) [결함 심각도]를 선택

구 분	내 용
치명적	고객에게 극단적인 고통을 주는 돌이킬 수 없는 결함
매우 심각	프로그램이 비정상적으로 끝나거나 리부팅 해야 하거나,중요한 기능이 다르게 동작하는 것
심 각	제품의 기능 또는 구성요소의 개념이 빗나가고 있지만, 피해갈 수 있는 방법이 있는 것(default)
보 통	기능이 다르게 동작하거나 어떤 특징이 없는 것
경 미	기능이 영향을 받지 않지만 바람직하지 않는 것, 예를 들면 화면에 위치 오류와 스펠링 오류 같은 것

(7) 기타 사항으로, 우선순위 선택, [릴리즈 계획]을 선택, [제목]/[요청내용]을 입력, 첨부할 파일 추가

DMT 처리시 입력되는 항목과 주요 항목별 상태 코드는 다음과 같다:

(1) 처리자가 최종처리자일 경우 처리결과를 선택

구 분	내 용
Progress	요청서의 처리 상태가 진행중(Default)
Approval	요청서의 처리 결과 상태가 승인
Non-Reproducible	요청서의 처리 결과 상태가 재현 불가능
Impossible	요청서의 처리 결과 상태가 처리 불가
ForceClose	요청서의 처리 결과 상태가 강제 종료

(2) 릴리즈 계획(요청 내용이 반영될 버전)을 선택하여 수정 가능

(3) 예상 처리 일자를 선택

(4) 처리자가 개발자일 경우 모듈명을 입력

(5) 다음 처리자를 선택(여러명 선택 가능)

(6) 해당 담당자의 답변을 작성

(7) [오류를 피해가는 방법], [최종 답변], [재현방법], [처리내역] [첨부 파일] 입력

[표 2]은 DMT 프로세스의 각 담당자 별로 그 역할에 따른 권한 및 관련 기능을 제시하고 있다.

[표 2] 역할별 기능 버튼 사용

구 분	QA (품질보증 담당자)	PM (프로젝트 관리자)	DEV (개발자)	BD (Builder)	SUB (Submitter)
승 인	0	0	0	0	0
회 신	0	0			
반 송		0			0
위 임			0		0
추 가			0		
프로젝트 변경	0	0			
FAQ 등록	0	0	0	0	0
저 장	0	0	0	0	0
강제 종료	0				0
보 류	0	0	0	0	0
삭 제	0				

3.3 Promise 시스템의 제공 정보

Promise 시스템은 개발과정에서의 결함을 줄이기 위한 인스펙션 활동, 테스팅 활동 등을 정량적으로 관리하기 위한 다양한 통계보고 기능을 제공합니다. 각 제품별, 릴리즈별, 기간별, 요청 구분별, 개발자별, 결과상태별로 등록현황, 진행현황, 주요통계 등이 제공되며, 그 세부 기능들은 [표 3]와 같다. 또한 화면으로 제공되는 모든 결과들은 Excel file로 export가 가능하여 필요한 통계자료의 추가 가공 등이 가능하도록 한다.

5. 결론 및 향후 연구 방향

본 논문에서는 기존의 결함 추적 시스템의 결점을 보완한 결함 및 요구사항 관리 시스템(Promise 시스템)의 요구사항과 개발 내용을 제시하였다. 현재 Promise 시스템은 다양한 통계 기능을 통해 각종 데이터 분석 활동을 지원하고, 제품의 요구사항을 체계적으로 관리하여 제품 기능 개선에 반영하며, 제품의 결함을 단계별 프로세스를 통해 관리함으로서 제품의 결함률을 감소시키고 수집된 데이터를 분석하여 그 결과를 제품에 피드백함으로서 체계적인 품질 관리를 할 수 있게 활용되고 있다.

[표 3] Promise 시스템의 주요 통계 정보

항 목		설명
등록 현황	날짜별 요청서 타입 현황	일자에 대해 RMT/DMT 요청서 타입별로 통계 결과
	날짜별 요청구분 현황	일자에 대해 요청 구분별 통계 결과
	날짜별 결함심각도 현황	일자에 대한 요청 등급별 통계 결과
	기능별 요청구분 현황	기능에 대한 요청 구분별 통계 결과
	기능별 우선순위 현황	기능에 대한 우선순위별 통계 결과
	기능별 결함심각도 현황	기능에 대한 요청등급별 통계 결과
	기능별 역할 현황	기능에 대한 역할별로 통계 결과
	개발자별 결함심각도 현황	개발자에 대한 요청등급별 통계 결과
	사이트별 요청구분 현황	사이트에 대한 요청 구분별 통계 결과
진행 현황	기능별 진행상태 현황	기능에 대한 진행 상태별 통계 결과
	CSR MyTodoList 현황	사용자에 대한 각 역할별로 CSR 요청서 MyTodoList의 통계 결과
	DMT MyTodoList 현황	사용자에 대한 각 역할별로 DMT 요청서 MyTodoList의 통계 결과
	개발자별 진행상태 현황	개발자에 대한 진행상태별 통계 결과
	요청자별 진행상태 현황	요청자에 대한 진행상태별 통계 결과
	결함심각도별 진행상태 현황	결함심각도에 대한 진행상태별 통계 결과
주요 통계	결함유입별 결함발견 현황	Where Found에 대한 Defect origin별 통계 결과
	버전별 릴리즈 이후 현황	버전별로 릴리즈 이후 오류/요구사항 현황에 대한 통계결과
	개발자별 평균처리시간 현황	개발자별로 평균 시간 현황 제공
	날짜별 반송처리 현황	날짜별로 반송 처리현황 제공
	단계별 결함유형 현황	결함발견 단계별로 결함 유형 현황 제공
	동료검토 보고서	동료검토 보고서를 조회
	Inspection 효과성	Inspection 효과성을 조회

특히, 경영진에게 제품별 월별 등록건수, 처리지연건수, 평균처리일수, 릴리즈 시 결함 수 등의 각종 통계 자료를 제공하여 릴리즈 검토, 테스트 활동 검토, 고객 대응 활동 등을 검토하고 하는데 활용되고 있다. Promise 시스템은 CMM level 3를 달성하는데 활용되었을 뿐 아니라 CMM level 5를 달성하는데 활용될 예정이다.

향후에는 BPM(Business Process Management) 도구를 전체 프로세스에 활용하여, 더욱 더 효과적인(agile and flexible) 결함관리 시스템을 구축하고자 한다. 일부 고객에게 이미 일부 기능을 BPM으로 구현하여 제공하였지만 사내 적용 및 판매를 위하여 더욱 개선하고자 한다. 그리고 현재는 데이터를 CSV파일로 내려 이를 excel등의 도구로 분석하고 있으나 향후에는 효과적인 reporting tool 과 연계하고자 한다.

참고문헌

[1] Promise 2.5 사용자 매뉴얼, 핸디소프트, 2004

[2] Bill Hetzel "The Complete Guide to software Testing" 2nd Ed. John Wiley & Sons, Inc. 1988.

[3] "CMMI-SE/SW/IPPD/SS Version 1.," http://www.sei.cmu.edu/cmmi/models

[4] Gaffney, J. , Cruickshank, R., Werling, R. and Felber, H. F., Software Measurement Guidebook, International Thomson Computer Press, Boston, 1995.

[5] IEEE Standard for Software Test Documentation, IEEE Std. 829-1983, NewYork; IEEE Press, 1983.

[6] http://linas.org/linux/pm.html

[7] http://testingfaqs.org/t-track.html

[8] Karl E. Wiegers(오세영 역), "Software Requirements 2/E(성공적인 프로젝트 수행을 위한 소프트웨어 요구사항)," Microsoft Press, 2003.

[9] Mark C. Paulk and et al., "The Capability Maturity Model: Guidelines for Improving the Software Process," Addison-Wesley Publishing Company, 1994.

[10] Matthew P. Barnson, "The Bugzilla Guide - 2.16.3Release,"http://www.bugzilla.org/docs216/html/, April 2003.

[11] PanKaj Jalote, "CMM in Practice: Processes for Executing Software Projects at Infosys," Addison-Wesley, 2000

[12] Watts S. Humphrey, "Managing the Software Process", Addison-Wesley Publishing Company, 1989.