DOI: 10.3745/KIPSTD.2008.15-D.3.321

결함 추적 시스템에 의한 소프트웨어 결함 분석 및 관리기법 연구

문 영 준† · 류 성 열††

요 약

프로젝트 진행중에 발견하지 못한 결함이 소프트웨어 개발 완료 후 유지보수 단계에서 발견되는 경우가 많이 있다. 유지보수 단계에서 결함의 발생 빈도가 높을수록 비용은 증가하고 품질은 저하되며 고객의 신뢰성을 떨어뜨린다. 결함은 조직에서 발생에 대한 원인 분석 및 프로세스 개선이 지속적으로 이루어지지 않으면 감소하지 않는다. 본 논문에서는 파레토 법칙에 따라 결함은 이미 발생된 유형이 반복되어 전체 결함 유형의 대부분을 차지한다는 점에 감안하여 DTS를 구현하였다. DTS는 유지보수 단계에서 과거에 발생했던 결함 유형의 이력을 바탕으로 결함의 원인을 추적하여 개발자, 운영자 및 유지보수 담당자에게 개선을 위한 근본 데이터를 제공함으로써 같은 유형의 결함이 반복적으로 발생하지 않도록 최대한 지원해 준다. DTS의 기본 활동은 프로그램의 결함유형 분석 및 측정 지표를 제공하고, 프로그램별 결함 유형을 집계한다. 이렇게 측정된 결함의 유형 사례를 해당 업무 팀에서 확인함으로써 지속적으로 결함을 개선할 수 있도록 지원한다. W사의 프로그램 형상관리 시스템에서 DTS를 구현하고 적용한 결과 약 65%정도의 결함이 개선되었다.

키워드 : 소프트웨어, 유지보수, 결함추적, 결함분석, 품질보증, 형상관리

A Study on Software Fault Analysis and Management Method using Defect Tracking System

Moon Young Joon† · Rhew Sung Yul††

ABSTRACT

The software defects that are not found in the course of a project frequently appear during the conduct of the maintenance procedure after the complete development of the software. As the frequency of surfacing of defects during the maintenance procedure increases, the cost likewise increases, and the quality and customer reliability decreases. The defect rate will go down only if cause analysis and process improvement are constantly performed. This study embodies the defect tracking system (DTS) by considering the Pareto principle: that most defects are repetitions of defects that have previously occurred. Based on the records of previously occurring defects found during the conduct of a maintenance procedure, DTS tracks the causes of the software defects and provides the developer, operator, and maintenance engineer with the basic data for the improvement of the software concerned so that the defect will no longer be manifested or repeated. The basic function of DTS is to analyze the defect type, provide the measurement index for it, and aggregate the program defect type. Doing these will pave the way for the full correction of all the defects of a software as it will enable the defect correction team to check the measured defect type. When DTS was applied in the software configuration management system of the W company, around 65% of all its software defects were corrected.

Key Words : Software, Maintenance, Defect Tracking, Fault Analysis, Quality Assurance, Configuration Management

1. 서 론

정보화 시대에 조직의 비즈니스 목적을 성공적으로 달성하기 위해서 빠르게 변화하는 비즈니스의 목적을 뒷받침하는 유지보수가 강력히 지원되어야 한다. 특히 소프트웨어 개발 후, 유지보수 환경에서 변화가 많이 발생하고 있으며 이에 대한 비용이 계속 증가하고 있고 유지보수 비용의 증가는 개발비용을 상회하고 있다.

특히 여러개의 시스템이 맞물려서 서로 중요 정보를 연동하고, 관리하는 대형 조직에서는 유지보수의 중요성이 심각하게 대두되고 있으므로 소프트웨어 시스템의 이해, 프로그램 이해와 설계, 시험 및 검증, 변경관리 및 문서화의 중요도가 크다[8].

※ 본 연구는 숭실대학교 교내 연구비 지원으로 이루어졌음.
† 정 회 원 : 숭실대학교 컴퓨터학과 박사과정
†† 종신회원 : 숭실대학교 정보과학대학 컴퓨터학부 교수
논문접수 : 2007년 8월 27일
수 정 일 : 1차 2007년 12월 17일, 2차 2008년 1월 16일, 3차 2008년 2월 26일
심사완료 : 2008년 2월 27일

대부분의 조직에서는 유지보수를 위한 기초 자료를 수집하고 원인을 분석하고 개선하는 활동이 없거나 미약한 경우가 많다. 이로 인하여 중복으로 투입되는 유지보수 비용이 증가되고 있다. 이를 개선하기 위하여 유지보수 환경에서 발생하는 결함을 추적하여 반복적인 오류가 발생하지 않도록 지원하는 시스템이 필요하다.

본 논문에서는 DTS(Defect Tracking System;결함추적시스템)을 구현하여 유지보수 환경에서 소프트웨어의 변경이 일어나는 경우 과거의 결함 정보를 측정 및 분석하고 추적관리를 통하여 과거의 유사한 오류가 발생하거나 반복적인 결함이 발생하는 소프트웨어의 변경시 통제를 강화하도록 하였다. 또한 결함이 발생된 후에는 원인을 피드백하여 개선할 수 있도록 제시하였다.

2. 관련 연구

2.1 유지보수 프로세스와 소요 공수

유지보수의 방법은 대상 업무의 개발 유형, 개인의 경력이나 경험 및 업무의 이해 정도에 따라 큰 차이를 보이고 있다. 유지보수 전담 조직을 가지고 있는 40개 업체를 대상으로 프로세스의 중요도, 오류율 및 소요공수를 산정한 결과는 <표 1>과 같다[9].

개발 프로세스에서는 방법론, 도구 등을 이용하여 요구사항에 근거한 충분한 업무 분석 및 설계, 테스트로 인하여 운영환경에 처음 적용할 때 안정적이다. 하지만 유지보수 프로세스에서는 업무요건이 자주 바뀌며, 처음에 개발에 참여하지 않은 사람이 유지보수에 투입될 경우 소프트웨어의 결함이 자주 발생한다. 업무 변경으로 프로그램의 수정이 불가피한 경우 업무의 변경요건을 명확히 이해하고 분석해야 한다. 변경 후, 오류가 발생하였다면 원인과 조치사항을 시스템에 기록하고, 향후 유사업무나 프로그램의 변경이 일어날 경우 과거 정보를 추적할 수 있어야 한다. 또한 실제 변경이 일어나기 전에 변경으로 인한 이상 효과를 점검할 필요가 있다[6].

2.2 SW-FMEA(Software Fault Mode Effect Analysis) 모델

소프트웨어의 품질 예방을 위한 방법으로 시스템의 안전

〈표 1〉 유지보수 프로세스와 소요 공수

프로세스	중요도	오류율	소요공수
사용자 요구사항 검토	2%	12%	5~8%
업무의 범위 파악	3%	5%	5~10%
일정 계획의 수립	5%	25%	3~5%
소프트웨어 시스템의 이해	15%	8%	10~20%
프로그램 이해와 설계	28%	7%	20~30%
코딩	1%	5%	5~10%
시험 및 검증	26%	32%	20~30%
변경관리 및 문서화	18%	0%	5~10%
기타	2%	6%	5~8%

확보를 위해 사용되고 있는 FMEA를 활용할 수 있다. SW-FMEA(Software Fault Mode Effect Analysis)는 예측을 통해 결함을 예방하는 방법으로, 기존에는 요구사항 분석 및 설계시 많이 활용되어 왔다.

이러한 SW-FMEA는 개발 활동을 통해 측정되는 정보를 활용하여, 분석, 설계, 나아가 동료검토나 테스트 등 개발 및 관리 활동에 적용하여 결함예방의 수단으로 활용 할 수 있다[10].

SW-FMEA 모델을 바탕으로 프로그램 개발단계에서 사전에 측정 정보를 활용하여 소프트웨어의 결함을 예방할 수 있다. 하지만 유지보수 단계에서의 소프트웨어 결함은 아주 단순한 규칙을 위배하여 발생하는 경우가 대부분이고 결함을 발견한 후, 대부분 10분 ~ 1시간이내에 원인을 찾아서 정상적으로 재등록 한다. 결함복구 시간동안에 지속되는 오류로 인하여 단순한 오류부터 치명적인 장애까지 발생되고 있으며 근본 원인을 찾아내어 결함을 최대한 예방할 수 있도록 시스템화 하여 관리하는 기법이 필요하다.

2.3 유지보수 스코어카드의 구조적 분석

MSC(Maintenance Scorecard)는 유지보수 단계에서 전략의 개발 및 수행과 전략적 목적 달성을 위하여 진행사항을 모니터링하거나 측정하고 관리할 수 있도록 측정지표를 만들어주는 틀이며 도구화가 가능하다.

유지보수 활동에서는 핵심 지표를 설정하여 원하는 측정이 무엇인지 이해하고 원하는 성능의 표준을 토론하고 측정과 결과를 상세화해야 한다. 특히 반복되는 실수로 발생하는 결함을 식별하고 측정해야 한다[2].

프로그램 결함의 개선을 위해서 핵심 지표를 결함율, 결함지연시간, 결함비용으로 정의하고, 특히 프로그램 결함에 있어서는 전체 유지보수 비용 대비 반복적인 실수로 인한 유지보수 비용을 측정하고, 반복적인 프로그램 결함이 발생하지 않도록 관리하는 프로세스가 필요하다.

2.4 Kurt Lewin의 역장분석

역장 분석(Force Field Analysis)은 사회심리학자 Kurt Lewin이 최초로 개발한 기법이다. 이 기법에서는 현재의 상황과 향후 달성하고자 하는 목표를 반영하여 목표 달성에 도움이 되는 요인과 방해가 되는 요인을 정리함으로써 긍정적인 힘과 부정적인 힘이 서로 밀고 당기면서 최선과 최악의 시나리오를 만들어내는 과정을 관찰할 수 있다[7].

6시그마 improve 단계에서 사용하는 기법으로 노력과 효과에 대한 수치는 최저 0점에서 최고 10점을 기준으로 평가하며 노력대비 효과 분석(Pay off Matrix)이다.

3. 결함추적시스템(DTS)의 구현

3.1 개발 환경

개발 후 유지보수단계에서 업무의 변화로 소프트웨어의 지속적인 개선이 필요하다. 유지보수 활동은 하드웨어나 운영체제의 변경으로부터 알맞게 대처하고 오류나 결함을 발

견하고 개선하여 시스템을 안정적으로 운영하는 것을 목표로 한다[1].

특히 유지보수단계에서 업무환경의 변화로 인한 불분명한 요구사항은 애매모호해서 여러 사람이 서로 다르게 이해할 수 있다. 그 결과로 개발자가 업무 로직을 요구사항과 다르게 구현하여 결함을 발생시킬 수 있다. 이를 예방하기 위하여 고객, 사용자의 업무에 대하여 분석가와 개발자를 교육시켜야 한다[5].

결함이 발생하면 소프트웨어 수정 후, 운영환경에 재등록한다. 이때 사후 관리를 위하여 결함의 원인과 조치사항에 대한 정보를 시스템에 등록하고 체계적인 관리가 필요하다. 또한 운영환경에서 동일 프로그램에 대한 반복적인 결함이 자주 발생하는 경우도 많이 발생하고 있다. 이러한 반복된 결함의 원인 데이터를 활용한다면 차후에 발생하는 결함을 예방할 수 있을 것이다. 결함추적시스템은 소프트웨어형상관리(SCM) 도구와 통합되어야 한다[3]. 이러한 시스템 환경을 구축하기 위하여 W사의 정보시스템은 122대의 프로그램 운영서버 및 메인 프레임(IBM,2084-307) 환경에서 프로그램 형상관리시스템(Program Configuration Management System)을 별도로 구축하였고 프로그램은 PCMS를 통하여 운영환경으로 이관된다. PCMS는 메인프레임과 게이트웨이를 통하여 연동되고, 122대의 각 서버와는 별도의 에이전트 소프트웨어를 개발하여 연동되도록 하였다. 조직의 모든 프로그램은 PCMS를 통하여 운영환경으로 이관되므로 모든 프로그램의 내용이나 산출물, 버전의 관리가 가능하다. DTS는 PCMS안에서 별도의 기능으로 구현되었으며 DB를 같이 공유한다. 개발 환경으로 언어는 JAVA 1.4 JSP를 사용하였으며, Web Application Server는 Tomcat, DBMS는 오라클, 운영체제는 유닉스를 사용하였다.

3.2 DTS 구축을 위한 요구사항 도출 및 효과 분석

3.2.1 DTS 요구사항 도출

역장분석에 의한 개선 세부사항 및 시스템 운영부서의 전략 목적 달성과 결함율 개선을 위하여 유지보수 자료에 근거하여 요구사항은 다음과 같이 도출하였다.

① 결함을 유형별로 분류하고 부서별 건수가 조회되도록 적용되어야 한다.
② 유형별 건수를 선택하면 결함의 원인 및 조치사항이 상세 조회 되도록 적용되어야 한다.
③ 등록하려는 프로그램을 추적하여 과거에 결함이 있었는지 조회되고 결함이 있었다면 결함율과 원인이 조회되도록 적용되어야 한다.
④ 등록하려는 업무에 대하여 과거에 오류가 있었는지 조회되고 오류가 있었다면 어떤 경우이고 어떻게 조치하였는지 조회되도록 적용되어야 한다.
⑤ 부서별 결함율을 측정하고 4시그마 수준(0.62%이하)이 유지되는지 조회되고 관리되도록 적용되어야 한다.
⑥ 결함율, 결함지연시간, 결함지연비용에 대한 측정지표를 만들고 측정할 수 있도록 적용되어야 한다.
⑦ 1년에 10회이상 등록하는 프로그램 중에 결함 발생이 3회 이상인 프로그램이 조회되고, 특별관리 하도록 적용되어야 한다.
⑧ 결함 발생 프로그램의 결함 원인을 분석하여 체크리스트에 반영하고, 등록시 체크리스트를 작성하여 첨부하도록 적용되어야 한다.

3.2.2 DTS 효과분석

결함을 추적, 분석 및 관리하기 위해서는 이를 시스템화한 DTS의 구축이 필요하며 요구사항을 바탕으로 문제를 해결하기 위하여 현 시스템의 문제점에 대한 근본 원인과 개선 대안을 도출하였다.

(그림 1)의 개선대안에 대한 설문내용은 시스템을 운영하면서 가장 취약하고 문제가 되었던 원인들에 대하여 브레인스토밍을 통하여 개선대안을 도출하고 12개를 선별하였다. 역장 분석결과 12개 항목 모두 노력대비 효과가 높게 나타났으므로 적극 추진 대상으로 선정되었다.

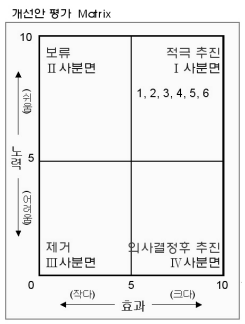
3.3 DTS 프로세스 및 아키텍처

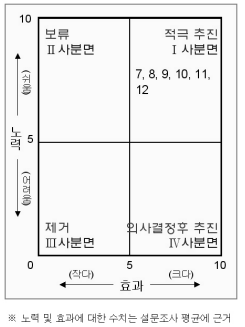
3.3.1 DTS 프로세스

DTS의 핵심 기능인 결함 추적은 개발자가 프로그램 수정 완료 후에 등록하는 시점에서 등록하려는 업무명이나 프로그램명을 등록의뢰서에 선택 또는 직접 입력하게 되고, 이러한 입력을 바탕으로 DTS에 저장했던 프로그램과 업무, 결함유형 간의 연결고리로 데이터를 추적하여 과거 결함이력이 있는 경우 프로그램 등록 전에 한번 더 체크하고 등록할 수 있도록 해준다.

Vital Few	NO	개선 대안	노력	효과
프로그램 등록 전 충분한 유형별 테스트 미흡	1	잠재원인 도출 유형별로 분류하여 프로그램 오류 유형 Case를 생성 한다. (PCMS 등록의뢰 시 유형 선택)	7	8
	2	결함 발생 건수가 많은 프로그램, 업무, 부서/개인에 대한 정보가 프로그램 참조 신청이나 등록의뢰 시 알림 기능이 필요하다.	6	9
	3	결함의 발생빈도가 높은 프로그램, 업무, 부서/개인에 대한 테스트 산출물을 자세히 검토한다.	8	8
프로그램의 오류를 사전에 예방 및 점검하는 체크리스트 미흡	4	프로그램 결함 발생 원인 및 조치결과 내용을 분석하고 반복적이며 공통된 요소를 찾아내고 결함 발생 유형을 사례별로 조사하고 체크리스트를 만든다.	6	8
	5	프로그램 등록 의뢰 시, PCMS가 판단하여 결함율이 높은 프로그램, 업무, 부서/개인이 의뢰할 경우 체크리스트를 첨부하라고 알려준다.	7	8
	6	체크리스트에 대한 지속적인 개선을 위하여 새로운 발생 유형에 대해 체크항목을 계속 추가하고 관리한다.	8	8

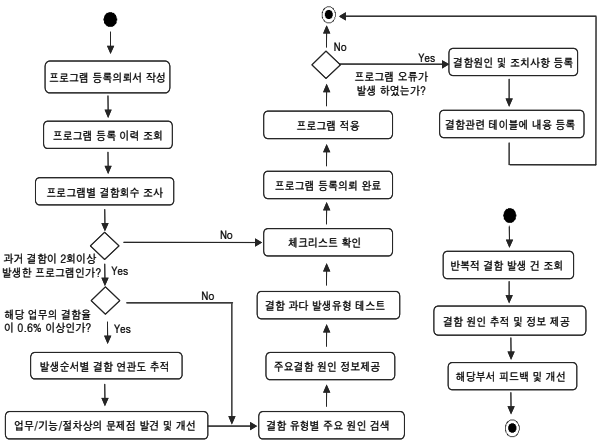
Vital Few	NO	개선 대안	노력	효과
프로그램 오류 발생 원인의 Feedback 이 부족	7	부서별/프로그램별 결함 분수를 집계하고 측정할 수 있는 지표를 체계화 한다.	8	8
	8	측정한 데이터의 정보를 관리부서 및 해당부서가 PCMS를 통하여 모니터링 할 수 있도록 한다.	9	9
	9	모니터링한 결과에 대하여 해당부서에 Feedback 하여 지속적인 결함축소가 이루어질 수 있도록 부서별 품질관리자가 주관하며 필요에 따라 활동한다.	6	9
요구사항 분석 및 업무의 이해도 부진	10	프로그램 결함 증에 원인이 업무에 대한 이해도 부족으로 발생한 결함에 대한 사례를 분류하고 공통원인 도출 후 PCMS 공지사항을 통하여 업무부서에 알려준다.	7	9
	11	결함 발생빈도가 높은 업무부서나 업무에 대하여 교육에 대한 방향을 제시하고 교육을 할 수 있도록 부서별 품질관리자가 주관하며 필요에 따라 활동한다.	6	9
	12	결함 발생 빈도가 높은 업무에 대하여 원활한 교육이 이루어 질 수 있도록 PCMS를 통하여 언제든지 모니터링 할 수 있도록 한다.	8	9





※ 노력 및 효과에 대한 수치는 설문조사 평균에 근거

(그림 1) 노력대비 효과에 의한 역장 분석 결과



(그림 2) DTS 프로세스

① 발생하는 결함의 원인 및 조치사항을 입력 받는다.
② 관리자는 ①번을 바탕으로 결함을 유형별로 분류하고, DTS는 이를 입력 받는다.
③ 프로그램 수정 시, 결함의 유형을 입력 받고, 유형별로 프로그램 본 수 단위로 집계한다.
④ 업무 부서별로 프로그램 등록대비 결함을 측정하고 이를 조회할 수 있도록 지원한다.
⑤ 결함 2회 이상 발생된 프로그램과 원인, 결함이 자주 발생하는 업무 정보를 저장한다.
⑥ 등록 의뢰 시, ⑤번에 해당되는 경우는 의뢰자에게 결함의 History를 제공한다.
⑦ 관리자는 반복되는 결함을 체크리스트로 만들고, DTS는 이를 의뢰자에게 제공한다.
⑧ 결함율, 결함지연시간, 결함지연비용에 대한 측정지표를 입력 받고 결과를 제공한다.
⑨ 목표수준(4시그마; 0.62%이하 수준)에 미달한 업무팀에 대하여 결함사례를 피드백 해준다.

위의 9가지 프로세스에 대한 흐름은 (그림 2)와 같다.

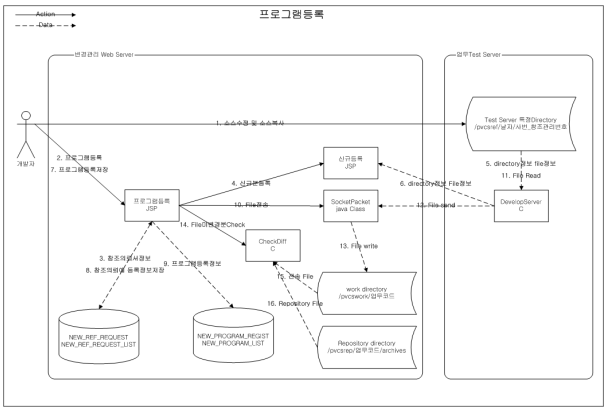
3.3.2 DTS 아키텍처

Java JSP, Tomcat, 오라클, 유닉스 환경에서 프로그램 등록시스템 환경에 포함되어 결함 축소를 위한 DTS가 구축되었으며 시스템 아키텍처는 (그림 3)과 같다.

4. DTS를 통한 결함 분석 및 운영 관리

4.1 프로그램 결함 유형 분석 및 측정

결함을 수집하고 분석하는 것은 개선을 위한 준비 과정이다. 조직의 프로그램 결함을 DTS에 축적하고 유형을 분석한다. 분석한 결함 유형의 원인 및 조치에 대한 이력을 찾아내고 시스템에서 선택적으로 결함의 유형을 추적한다. 정량적 관리를 위하여 결함 축소를 위한 측정지표를 만들고 계산식에 의해서 집계되며 시스템에 의하여 지속적으로 관리된다. 조직의 품질관리 부서는 DTS의 측정지표 결과를



(그림 3) DTS 아키텍처

〈표 2〉 측정 지표

측정지표	계산식	Spec/Defect 기준
전체 프로그램 결함율	{ ∑ 프로그램 결함 수 / ∑ 프로그램 등록 수} * 100	프로그램 결함율 4시그마 수준인 0.6% 이하 수준 목표
프로그램 결함 지연 시간	∑{∃x [결함 프로그램 개선 시점 - 결함 프로그램 등록 시점]}	
프로그램 결함 지연 비용	∑{프로그램 결함 지연 시간 * (∀x [인건비] / 직원 수)}	

바탕으로 결함 유형별 결함 발생 건 수를 부서별로 조회한다. 결함 발생 비율이 4시그마 수준인 0.6%가 넘는 부서에 대하여 결함유형 및 관련 프로그램을 DTS에서 확인하고 원인을 찾아내어 피드백을 주고 개선하도록 요청한다.

4.2 DTS의 운영 관리

4.2.1 DTS 의 관리 방법

4.2.1.1 DTS의 관리 프로세스

조직에서는 새로운 업무 환경에 대비하기 위하여 프로그램을 지속적으로 개선하고 Real 환경에 반영하고자 프로그램 형상관리 시스템을 통하여 프로그램을 등록 요청한다. 이때 프로그램 QA 담당자는 파레토법칙에 의하여 오류를 자주 발생시키는 업무 팀이나 개발자를 주시하여 시스템에 기록된 History를 모니터링 하여 프로그램 결함으로 인한 등록율이 1%이상 되는 업무 팀 및 담당자의 프로그램 등록건에 대한 QA 확인을 강화한다. 특히 Soft Copy로 문서화하여 프로그램 형상관리 시스템에 첨부한 테스트 자료를 면밀히 검토 한다. 또한 프로그램 등록의뢰서의 테스트 내용 및 업무내용 상세 점검과 산출물 확인을 강화한다.

4.2.1.2 테스트 체크리스트 작성 및 확인

QA는 품질의 효과성과 완전성을 제어하는 활동으로 감사와 기능별 보고로 구성된다[4]. 프로그램 QA 담당자는 결함 유형별 사례에 대한 테스트 체크리스트 작성하고 업무팀에서 프로그램 등록 시 체크하였는지 확인한다. 테스트 체크리스트는 프로그램 형상관리 DB에 축적된 결함의 유형을 분석한 후 유사한 결함단위로 나누어 체크리스트를 만들고 개발자에게 배포한다. 이로써 개발자는 과거에 반복되었던 오류를 다시 범하지 않도록 체크 할 수 있다. 특히 부서

별로 결함의 발생빈도 유형이 다르게 나타날 수 있으므로 부서에 맞는 특정적 체크리스트와 전체적으로 공통된 일반적 체크리스트로 분류하여 배포한다.

4.2.1.3 업무 결함 유형 분석 및 개선 조치 요구

프로그램 결함의 40% 이상이 업무에 대한 요구사항 파악이 미흡하거나 업무 프로세스의 인식 부족 등으로 발생한다. 각 부서별로 업무 미숙으로 발생하는 결함의 유형을 분석하고 관련 업무별로 분류한다. 이에 대한 시스템 분석자료를 업무 팀에서 Feedback 받고, 필요에 따라 관련자 교육 및 부서장의 주관 하에 훈련을 통하여 효과적인 개선이 이루어 질 수 있도록 한다. 품질관리 조직에서 결함 유형별 상세내역 조회 및 시정 요구를 하면 해당 부서에서는 업무에 대한 미숙으로 발생한 업무 로직 결함 건에 대하여 업무 숙지를 강화하고 주의할 것이다.

4.2.2 DTS 의 운영 결과

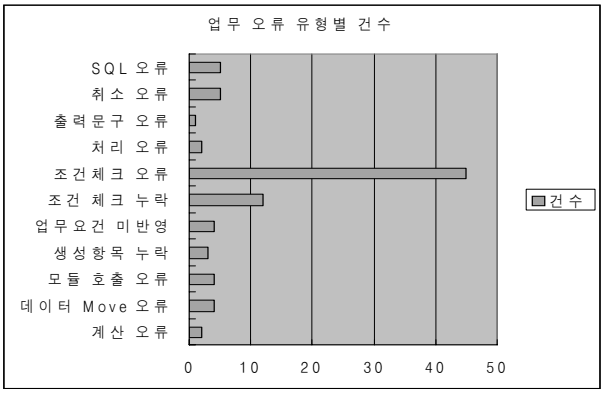
DTS의 결함 등록현황 조회는 (그림 4)와 같이 결함 유형을 세분화하여 기간별, 부서별 결함수를 조회할 수 있으며 해당 결함수를 클릭하면 상세내역 조회가 가능하다. 이로써 빈도가 높은 결함 유형을 한눈에 쉽게 파악할 수 있고 해당 유형의 결함을 상세히 조회할 수 있다.

W사에 대하여 DTS를 적용한 결과를 보면 업무 로직관련 오류가 전체의 44%를 차지하고 있으며 이를 축소하면 결함율을 많이 떨어뜨릴 것이다. 업무 로직으로 인한 프로그램 결함 건을 자세히 분석하면 계산 오류, 데이터 Move 오류, 모듈 호출 오류, 생성항목 누락, 업무요건 미반영, 조건 체크 누락, 조건체크 오류, 처리 오류, 출력문구 오류, 취소 오류, SQL 오류로 분류되며 (그림 5)와 같다.

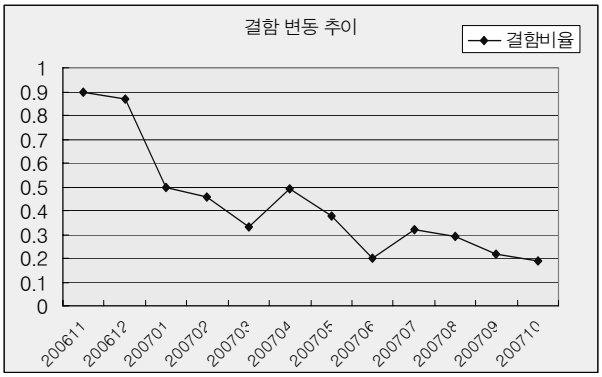
(그림 5)에서 보듯이 업무 오류의 65.5%가 조건 체크 누락이나 오류로 발생하였다. 이러한 부분을 더욱 개선한다면 프로그램 결함율은 현저히 떨어질 것이다. 또한 사후관리 차원에서 소프트웨어 결함의 원인 및 조치사항, 반복적으로 발생하는 오류의 원인 등의 정보를 해당부서에 통보하고 시정조치 하도록 하여 지속적인 개선이 이루어지도록 한다. 시험 운영결과 이러한 노력과 지난 1년간 DTS에 의한 지원으로 소프트웨어 결함율이 (그림 6)과 같이 감소함을 볼 수 있다. 결함율을 전년도와 비교하면 0.94%에서 0.33%로 약 65% 정도 개선되었다.

5. 결론 및 향후 연구

본 논문에서는 개발후 운영환경에서 사용되다가 변경으로 인한 결함이 발생하는 경우 결함정보가 시스템에서 관리되고, 수집되는 데이터를 측정하고 이러한 정보를 이용하여 소프트웨어의 결함을 축소하는 방법을 제시하였다. 파레토 법칙에 따라 조직의 프로그램 오류나 결함발생이 특정 조직 및 특정 업무 개발자에 집중되어 있다. 즉, 이미 발생한 유형이 반복적으로 발생하여 전체 결함의 대부분을 차지하는 원리를 찾아 볼 수 있었다. 이러한 개념을 바탕으로 반복되는 결함을 제거하기 위하여 DTS를 구축하였고, 추이를 보면서 자주 발생하는 조직의 결함 유형을 찾아내었다. 그 후, 결함의 유형을 지속적으로 측정하였고 개선 차원에서 조직의 관련 업무 담당자에게 피드백을 해주었다. 또한 조직의 품질 담당자는 형상관리 시스템의 프로그램 오류나 결함을 유형별로 분석하여 자주 발생되는 원인들을 체크리스트로



(그림 5) 업무 로직 오류의 유형별 건수



(그림 6) 결함율 개선 결과

프로그램 오류 유형별 집계표 (2007-01-01 ~ 2007-12-31)

조회기간 2007-01-01 ~ 2007-12-31 시스템 전체 검색 엑셀

팀명\유형	오류(1)	오류(2)	오류(3)	오류(4)	오류(5)	오류(6)	오류(7)	오류(8)	합계
A팀	5 (0.39%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	1 (0.08%)	0 (0%)	0 (0%)	6 (0.47%) / 1269
B팀	12 (0.07%)	9 (0.05%)	3 (0.02%)	1 (0.01%)	1 (0.01%)	1 (0.01%)	0 (0%)	0 (0%)	27 (0.16%) / 17235
C팀	10 (0.08%)	7 (0.05%)	3 (0.02%)	2 (0.02%)	1 (0.01%)	4 (0.03%)	0 (0%)	16 (0.12%)	43 (0.33%) / 13168
D팀	2 (0.02%)	3 (0.03%)	0 (0%)	0 (0%)	0 (0%)	5 (0.04%)	1 (0.01%)	0 (0%)	11 (0.09%) / 11685
E팀	2 (0.03%)	1 (0.01%)	0 (0%)	0 (0%)	0 (0%)	2 (0.03%)	0 (0%)	0 (0%)	5 (0.07%) / 6873
F팀	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	1 (0.17%)	2 (0.33%)	0 (0%)	3 (0.5%) / 601
G팀	55 (0.41%)	8 (0.06%)	3 (0.02%)	1 (0.01%)	8 (0.06%)	9 (0.07%)	5 (0.04%)	9 (0.07%)	99 (0.74%) / 13428
H팀	25 (0.14%)	2 (0.01%)	4 (0.02%)	6 (0.03%)	10 (0.06%)	10 (0.06%)	1 (0.01%)	7 (0.04%)	65 (0.38%) / 17316
I팀	23 (0.39%)	17 (0.28%)	6 (0.1%)	2 (0.03%)	4 (0.07%)	21 (0.35%)	0 (0%)	8 (0.13%)	81 (1.35%) / 5967
J팀	4 (0.02%)	8 (0.03%)	1 (0%)	1 (0%)	4 (0.02%)	7 (0.03%)	3 (0.01%)	7 (0.03%)	35 (0.14%) / 24146
K팀	7 (0.13%)	2 (0.04%)	1 (0.02%)	2 (0.04%)	4 (0.07%)	1 (0.02%)	1 (0.02%)	5 (0.09%)	23 (0.42%) / 5423
합계	145 (0.12%)	57 (0.05%)	21 (0.02%)	15 (0.01%)	32 (0.03%)	62 (0.05%)	13 (0.01%)	52 (0.04%)	398 (0.33%) / 120440

* 오류 유형별 내용

오류(1) - 로직 오류(업무처리 관련)	오류(2) - 로직 오류(시스템기능 관련)	오류(3) - DB 입력된 Data 오류
오류(4) - 테스트와 운영환경의 불일치 오류	오류(5) - Program/DB Layout 조합 오류	오류(6) - 프로그램 일부 미적용 오류
오류(7) - 타 System/Program 연동 및 프로세스 미준수 오류	오류(8) - 기타 오류(기타 사유 입력건)	합계 - 오류(1)~오류(8)

(그림 4) 프로그램 오류 유형별 집계표

작성하고 관련 부서에 배포하였으며, 새로운 결함 유형이 나타나면 계속 업그레이드 하였다.

DTS는 결함이 자주 발생하는 부서나 담당자가 관리하는 소프트웨어에 대하여 결함이 높은 프로그램을 분석하고 알려주어, 한번 더 점검하며 체크할 수 있도록 지원해준다. 이렇게 조직의 업무 특성에 맞는 프로그램 결함의 원인을 분석하고 결함의 잠재원인을 미리 예측하는 DTS를 구축하여, 결함 발생 후 들어가는 유지보수 비용을 절감하였고 소프트웨어의 신뢰성을 확보하여 고객이나 사용자의 만족도를 높일 수 있었다. 이러한 방법으로 조직의 유지보수를 지속적으로 개선한다면 불필요한 결함 수정 시간을 절약하고 고객으로부터의 신뢰도가 높아져서 조직의 성숙도가 향상되고 품질이 개선될 것이다.

참 고 문 헌

[1] Penny Grubb, Armstrong A Takang, Software Maintenance Concepts and Practice, World Scientific, 2003.

[2] Daryl Mather, The Maintenance Scorecard, Industrial Press, 2005.

[3] Matthew B. Doar, Practical Development Environments, O'Reilly & Associates, 2005.

[4] Roger S. Pressman, Software Engineering A Practitioner's Approach, McGraw-Hill, 2005.

[5] 칼위거스, 소프트웨어 요구사항, 정보문화사, 2003.

[6] 최은만, 소프트웨어공학, 정익사, 2007.

[7] 심현택, 6시그마란 무엇인가?, 창현출판사, 2002.

[8] 카네기멜론대학, S/W 개발 프로세스를 개선하기 위한 역량 성숙도 모델,피어슨 에듀케이션 코리아, 2002.

[9] 류성열, 백인섭, 김하진, "유지보수 관리 체계의 정형화 및 비용 예측 모델에 관한 연구," 1996년도 한국정보처리학회 논문지, pp.846-854.

[10] 김효영, 한혁수, "SW-FMEA 기반의 결함 예방 모델," 2006년도 한국정보과학회 논문지, pp.605-614.

문 영 준

e-mail : infolove@ssu.ac.kr

2004년 한국방송통신대학교 컴퓨터과학과 (학사)

2006년 연세대학교 컴퓨터공학과 (공학석사)

2006년~현 재 숭실대학교 컴퓨터학과 박사과정

관심분야 : 소프트웨어 유지보수, 형상관리, QA, UML, CBD, 요구공학, 정보보호

류 성 열

e-mail : syrhew@ssu.ac.kr

1976년 숭실대학교 전자계산학과(학사)

1980년 연세대학교 전자계산학과 (공학석사)

1997년 아주대학교 컴퓨터공학과 (공학박사)

1982년~1995년 숭실대학교 전자계산연구소 및 중앙전자계산소 소장

1997년~1998년 George Mason University 객원 교수

1998년~2001년 숭실대학교 정보과학대학원 원장

1998년~현 재 숭실대학교 소프트웨어 공학센터 소장

2006년~현 재 공정거래위원회 성과관리위원회위원

1981년~현 재 숭실대학교 정보과학대학 컴퓨터학부교수

관심분야 : 리엔지니어링, 소프트웨어 유지보수, 소프트웨어 재사용, 소프트웨어 재공학/역공학, 소프트웨어 테스팅