CMMI 성숙도 단계별 특징을 반영한 결함 관리 프로세스

한동준* · 조성민** · 이용섭*** · 한혁수****

1. 서 론

고객에게 인도된 소프트웨어 제품이 고객의 요구사항과 일치하지 않는다면, 그 제품은 오류/장애/실패(통칭하여 결함(defect))가 많이 내포되어 있다는 의미가 된다. 품질이 좋은 소프트웨어는 이러한 결함이 없이 고객의 요구사항과 일치하는 소프트웨어를 말한다. 개발 초기에, 보다 효과적으로, 그리고 체계적으로 결함을 관리할 수 있다면 많은 소프트웨어 프로젝트는 품질이 좋은 소프트웨어를 만들 수 있을 것이다. 이를 위해 품질에 많은 영향을 끼치는 결함을 효과적으로 관리할 수 있는 결함 관리 프로세스(Defect Management Process)와 결함 관리 시스템(Defect Management System)의 도입이 필요하다. 기존의 결함 관리 시스템들은 결함 관련 데이터를 한꺼번에 수집하기만을 요구하였다[1]. 즉 해당 데이터를 왜 모아야 하는지에 대한 인식이 부족한 상태에서 데이터를 수집하기만을 요구하여, 제공된 데이터의 신뢰성에 문제점을 야기했다. 그리고 결함 관리 프로세스가 준비되지 않은 조직은 프로세스에 의해 시스템을 운영하지 않으므로 결함 관리 시스템에 의해 수집된 결함 관련 데이터를 관리하기가 어렵다. 또한 결함 관리 프로세스를 다루는 모델이나 표준을 기반으로 하는 지침이 일반화 되지 않아 결함 관리 프로세스를 구축하는 것이 어렵다.

결함 관련 표준인 IEEE Std 1044에서는 결함의 유형을 정의하고 결함을 제거하는 리뷰, 테스팅의 활동만 정의하고 있어 실제 결함 관리 프로세스를 구축하는데 큰 도움이 되지 않는다[1]. 또한 CMMI에서도 성숙도 수준 2에 직접적으로 결함 관리 활동을 수행하는 프로세스 영역을 가지고 있지 않다[2]. 그러나 대부분의 소프트웨어 개발 조직은 테스팅을 통해 결함을 제거하고 있으며, CMMI 기반의 조직들도 이미 성숙도 수준 2에서부터 테스팅과 디버깅을 통해 결함을 제거하고 있다[3]. 이에 본 논문에서는 CMMI 기반의 결함 관리 프로세스를 개발하고, 이를 지원하고 기존의 결함 관리 시스템을 도입하였을 때 생기는 문제점을 해결한 개선된 결함 분석 및 통제 시스템을 제공하고자 한다. 그리고 본 논문에서 제안한 관리 방식과 시스템을 적용하면 조직들은 CMMI를 기반으로 어떤 결함 데이터를 모아야 하는지, 어떻게 결함 관리를 해야 하는지에 대한 지침을 제

※ 교신저자(Corresponding Author) : 한혁수, 주소 : 서울시 종로구 홍지동 7(110-743), 전화 : 02)2287-5033, FAX : 02)2287-0023, E-mail : hshan@smu.ac.kr
* 상명대학교 일반대학원 컴퓨터과학과 석사과정 (E-mail : lexia@smu.ac.kr)
** Teradata 컨설턴트 (E-mail : upoi@smu.ac.kr)
*** 상명대학교 일반대학원 컴퓨터과학과 석사과정 (E-mail : seotaiji@smu.ac.kr)
**** 상명대학교 소프트웨어학부 교수

공받게 된다.

2. CMMI

CMMI(Capability Maturity Model Integration)는 과거 소프트웨어 프로세스 개선 모델로써 사용되던 SW-CMM의 진화된 모델로서 SW-CMM V2.0 Draft C, EIA/IS 731, IPD-CMM V0.98을 기반으로 개발되었다. CMMI는 과거 SW-CMM이 소프트웨어와 관련된 프로세스 개선 활동만을 지원하던 취약점을 보완하여 소프트웨어, 시스템, 하드웨어, 서비스와 관련된 모든 프로세스 개선 활동을 지원한다[4]. CMMI의 성숙도 표현 방식은 그림 1과 같이 CMM과 같은 5개 성숙도 수준으로 나누어지며 각 성숙도 수준은 해당 수준에서 수행해야 하는 22개의 프로세스 영역(Process Area)을 제시하고 있다[3].

3. 결 함

3.1 결함이란?

결함의 정의는 그림 2와 같이 품질 관련 모델이나 전문가들마다 다양하다. 결함은 오류/장애/실패를 포함하며 고객이 요구하고 의도한 것과 산출

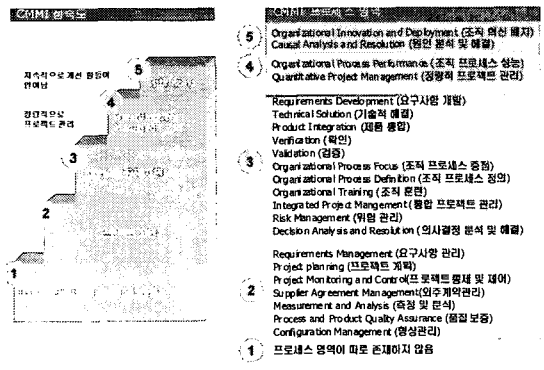

그림 1. CMMI 단계 및 프로세스 영역

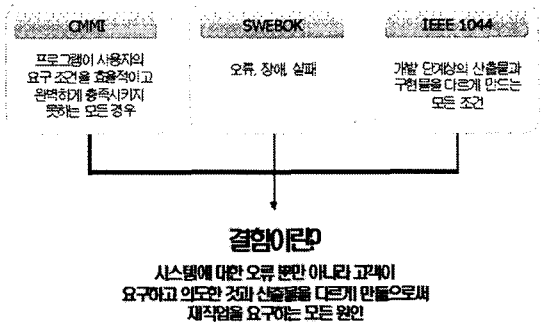

그림 2. 결함 정의

물을 다르게 만듦으로써 재작업을 요구하는 모든 원인으로 정의 된다[5]. 따라서 결함을 방지하고, 예방하는 활동이 소프트웨어의 품질을 좌우한다. 그러므로 여러 의미에서 결함 데이터는 공수 데이터보다 더 중요하다[6]. 우선, 결함 데이터는 프로젝트 관리를 위해 필요하다. 대규모 프로젝트는 수 천 개의 결함을 포함할 수 있는데, 이 결함은 각기 다른 사람들이 프로젝트의 각각 다른 단계에서 발견하게 된다. 흔히 프로세스에서 결함을 고치는 사람과 결함을 발견하거나 보고하는 사람은 서로 다르다[7].

3.2 결함 생명주기

결함 생명주기는 아래 그림 3과 같다[8].
이러한 결함 생명주기에 영향을 주는 부서는

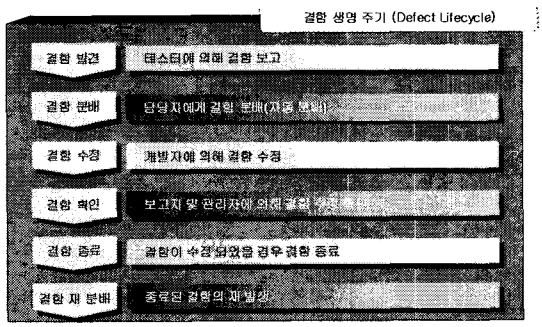

그림 3. 결함 생명 주기

크게 결함 보고 부서와 결함 수정 부서이다. 결함 수정 부서는 보고된 기능 오류를 검토하고 해결하는 작업을 수행한다. 반면, 결함 보고 부서는 결함 발견 및 분배, 개발자에 의해 해결된 결함의 수정 확인, 수정되지 않은 결함 재분배, 수정된 결함 종료 등의 작업을 수행한다.

여기서 "결함 확인"은 발견된 결함이 개발자에 의해 보고된 오류가 수정되어, 수정 사항이 적용된 새로운 제품에서 확인했을 때 문제가 수정되었음을 확인한 상태를 의미한다. "결함 종료"는 제품 테스트를 종료하는 최종 단계에서만 표시할 수 있는 오류의 상태이다. 테스트 최종 제품에서 지금까지 보고된 오류 중 수정된 오류, 즉 "결함 확인" 상태의 오류를 재현해보고기존에 수정된 결함이 모든 운영 시스템(OS)에서 발생하지 않고 시스템에 다른 문제를 일으키지 않는지 확인한 후 이상이 없을 경우에만 오류의 상태를 "종료"로 변경할 수 있다. 이는 회귀 테스트(Regression Test) 과정에서 예기치 못한 결함이 발생하여 종료한 결함을 다시 "재분배" 해야 하는 경우가 많기 때문이다. 이렇게 함으로써 개발자나 형상 관리 담당자가 파일 관리에 있어서 잘못된 파일(오류 수정 이전의 파일)이나 오류를 수정하면서 다른 오류를 발생하게 하는 실수를 감지해낼 수 있다.

이외에도 "연기(Defer)" 상태가 존재하며, 이는 경영진과 개발/QA 조직 모두가 중요하지 않은 문제이거나또는 현재 해결할 수 있는 방안이 없어서 다음으로 수정을 미룰 때를 의미한다. "중복(Duplicate)" 은 이미 생성되어 있는 결함을 반복하여 생성하였을 때를 의미한다.

이 밖에도 "정보 부족 (NMI - Need More Information)", "결함 아님 (NAB - Not A Bug)", "재현 불가(NREP - Not Reproducible)", "수정 계획 없음(NPTF - No Plan To Fix)", "사용자 오류(User Error)" 등의 상태가 존재한다[5]. 임무 분리 특성과 주체나 객체의 보안특성에 따라 접근을 통제함으로써 강화된 보안기능을 제공한다.

3.3 결함 제거 및 예방

3.3.1 결함 제거

결함이 발생하는 상황에서 최선의 목표는 최소한의 결함을 가진 소프트웨어를 인도하는 것이다. 이 목표를 달성하려면, 우선 인도하기 전에 가급적 모든 결함을 제거해야 한다. 프로젝트의 프로세스에는 결함을 발견하고 제거하기 위한 많은 활동들이 내포되어 있다. 비록 발견과 제거가 두 개의 고유 활동이지만, 두 가지를 함께 참조하기 위해 '제거(Removal)'라는 용어를 사용할 것이다. 결함의 잠복기가 길수록 제거 비용이 더 많이 든다[8]. 따라서 성숙한 프로세스는 결함이 주입될 만한 각 단계의 끝부분에 품질 통제 활동을 두게 될 것이다. 결함 제거 활동은 요구사항 검토와 설계 검토, 코드 검토, 단위 테스트(UT: Unit Testing), 통합 테스트(IT: Integration Testing), 시스템 테스트(ST: System Testing), 그리고 수락 테스트(AT: Acceptance Testing)를 포함하고 있다.(계획 문서에 대한 검토가 소프트웨어의 품질을 개선하는데 도움이 되지만, 여기에서는 포함시키지 않았다.) 그림 4에서는 결함 주입과 제거 프로세스를 보여주고 있다. 품질 관리 작업은 적합한 품질 통제 액티비티를 계획하고, 이 액티비티를 정확히 수행하고 통제하여 모든 결함을 소프트웨어가 고객에게 인도되기 전에 '프로세스 안(in-process)'에서 발견하는 것이다[9].

3.3.2 결함 예방

결함 예방(Defect Prevention)이란, 이미 주입된 결함 또는 잠재된 결함의 식별, 새로운 결함이 제품에 유입되는 것을 예방하는 활동을 의미한다

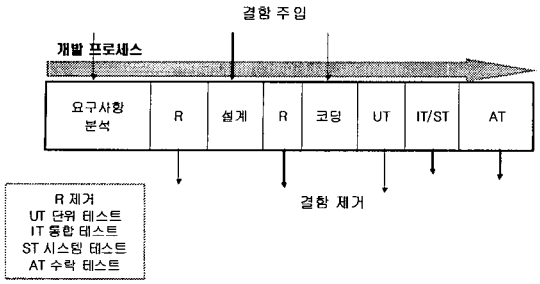

그림 4. 결함 주입과 제거

[10]. 이러한 결함 예방의 목적은 결함의 근본 원인을 제거하여, 결함의 유입을 사전에 방지하는 것이다. 결함 예방은 구조적 활동과 분석적 활동을 통해 수행된다. 구조적 활동은 적절한 프로세스(Process), 방법, 가이드라인을 통해 결함의 주입과 발생을 예방하는 것이며, 분석적 활동은 결함의 근본 원인 분석을 통해 여러 가지 리스크로부터 결함을 예방하고 재발을 방지하는 것이다. 예를 들어 데이터를 수집하고 분석하는 활동은 분석적 활동으로, 분석된 결과를 프로세스와 가이드라인 등의 형태로 개발 프로세스에 반영하는 활동은 구조적 활동으로 설명될 수 있다[4]. 결함 예방 활동에는 결함 제거 활동이 포함되어 있지만, 엄밀히 말하면 결함 제거와는 다르다. 결함제거 활동은 단일 프로젝트 내에서 개발되는 산출물이나 정보, 그리고 특정 단계에서 주입되는 결함들을 찾는데 중점을 두고 있다. 예를 들어 코딩 표준은 근본적으로 결함 주입을 예방하는 활동으로 볼 수 있고, 코딩 표준이 반영된 코드 리뷰(Code Review) 체크리스트의 활용은 유입된 결함을 조기에 발견하는 결함 제거 활동이라 볼 수 있다.

3.4 결함 관리 시스템

결함 추적 시스템(Defect Tracking System)이라고 불리기도 하는 결함 관리 시스템(Defect ManagementSystem)은 소프트웨어 프로젝트를 진행하면서 발생되는 결함을 관리할 수 있도록 도구 형태로 제공되는 것으로 결함 보고자와 결함 수정자간의 의사소통을 원활히 할 수 있도록 도와준다. 이러한 도구는 데이터베이스(Database)를 기반으로 결함을 데이터베이스에 레코드(Record) 형태로 입력(생성)/수정/삭제하면서 결함 상태에 따라 결함을 관리하는 시스템이다[5]. 여기서 결함(Defect)은 버그(Bug), 오류(Error), 해결 대상(Issue) 등과 혼용하여 사용하고 있다. 같은 맥락에서 결함 관리 시스템은 버그 추적 시스템(Bug Tracking System)이라고도 한다. 버그 추적 시스템은 Mantis, Bugzilla, Tracker, Debian Bug Tracking System, Open Track, Time Sheets for Networks(TSN), journyx Time 등의 수 많은 공개/free 소프트웨어와 TestTrack Pro, Rubicon Tracker, Keystone 등의 여러 상용 소프트웨어가 있다[6][11-13]. 결함 관리 시스템의 주요 기능에는 결함의 키워드 검색과 테스터의 시스템 사용 권한 관리 등이 있다[14]. 일반적인 소프트웨어 결함 관리 시스템에서 제공되는 공통된 데이터를 정리하면 다음 표 1과 같다.

표 1. 결함 데이터

번호	결함 데이터	번호	결함 데이터
1	결함번호	11	날짜 (등록/수정/빌드)
2	요약기술	12	결함을 수정하는 사람
3	제품/프로젝트 이름	13	결함에 대한 해결책
4	제품 버전	14	할당된 QA
5	수정된 빌드 번호	15	처리상태
6	결함을 만들기 위한 과정	16	우선순위
7	간단한 설명	17	심각도
8	결함을 신고한 사람	18	히스토리
9	테스트 환경	19	파일첨부
10	카테고리		

본 논문에서 결함은 버그, 오류 그리고 해결 대상 등을 모두 포함하여 사용한다.

3.5 기존 결함 관리 시스템의 문제점

결함 데이터를 처음 모으거나 결함 및 측정에 대한 이해가 약한 조직은 기존 결함 관리 시스템에서 요구하는 데이터들을 체계적으로 수집할 수 없다. 시스템 도입에 따라 많은 데이터를 한꺼번에 모음으로써, 해당 데이터를 왜 모아야 하는지에 대한 인식이 부족하고 결함 관리 프로세스 없이 도구만 도입함으로써 결함 관련 활동을 조직에 정착시키지 못했다. 또한 결함 관리 시스템을 통해 나온 결함 데이터를 어떻게 분석해야 이후 프로젝트에 도움이 될 수 있을지에 대한 지침도 없다. 이에 본 논문은 결함 관리 프로세스와 이를 지원하는 결함 관리 도구의 개발을 통해 이러한 문제점을 해결하고자 한다.

4. CMMI 기반의 결함 관리

아래의 그림 5와 같이, 결함 방지 활동과 관련한 프로세스 영역은 CMMI 성숙도 수준 5의 CAR(Causal Analysis and Resolution)에 명시되어 있다. CMMI 성숙도 수준 2의 MA(Measurement and Analysis)와 CMMI 성숙도 수준 3의 VER(Verification)과 VAL(Validation) 그리고 CMMI 성숙도 수준 4의 OPP(Organizational Process Performance), QPM(Quantitative Project Management)는 결함 관리 활동에 필요한 프로세스 영역이다.

CMMI 성숙도 수준 2의 7가지 프로세스 영역에는 결함 관리의 가장 큰 활동인 '발견'과 '제거'에대한 프로세스 영역은 존재하지 않는다. 결함의 '발견'과 '제거'에 관한 프로세스 영역은 성숙도 수준 3의 VER(Verification)과 VAL(Validation) 프로세스 영역에서 수행된다. 그러나 검증 및 확인 프로세스 영역을 수행하자 않는 소프트웨어 개발 조직도 테스팅을 통해 결함을 발견하고 제거한다. 따라서 CMMI의 특정 성숙도에 도달한 조직이 아니라도 성숙도 수준 2의 프로세스 영역인 MA(Measurement and Analysis)를 기반으로 하여 결함 관리를 시작할 수 있다. MA는 그림 6과 같이 결함 관리라는 목표 아래 측정 메트릭(Metric)을 선정하고, 이를 위한 데이터를 수집하도록 요구한다.

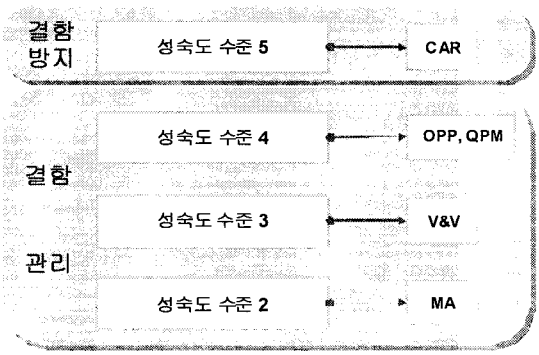

그림 5. CMMI와 결함 관련 PA

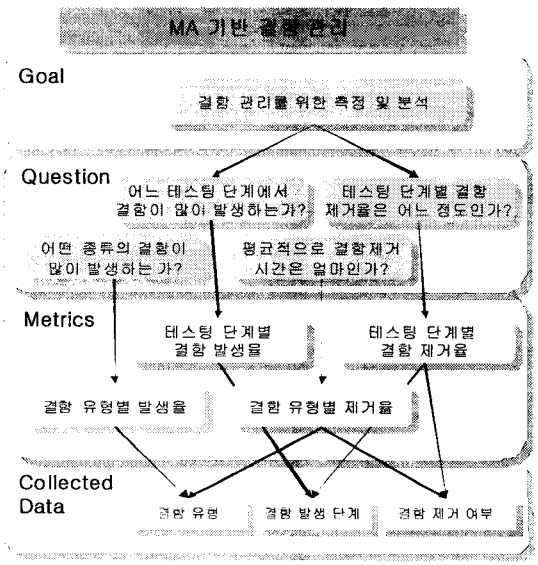

그림 6. CMMI 성숙도 수준 2의 MA기반 결함 관리

이와 같이 성숙도 수준 2기반의 결함 관리 목표는 결함을 발견하여 제거하고, 결함 관련 데이터를 수집하여 결함의 유형별 발생, 발생 빈도 수, 결함 분포 등의 정보를 얻는 것이다[15].

CMMI 성숙도 수준 3의 특징은 해당 조직에서 따라야 하는 조직 차원의 표준 프로세스를 보유하고 있는 것이다. 이러한 조직 표준 프로세스는 해당 조직에서 프로젝트를 수행할 경우 프로젝트 특성에 따라 적절하게 조정하여 사용된다[2].

이러한 성숙도 수준은 훨씬 구체적이고 정교한 프로세스를 제공하기 때문에 성숙 수준 2 기반의 결함 관리와 차별화 된다[3]. 소프트웨어 개발 조직은 이러한 구체적인 프로세스를 이용해 측정치를 수집할 수 있으며, 또한 프로세스들 간의 관계를 정확하게 이해할 수 있다. 이러한 특성에 비추어 성숙도 수준 3에 해당하는 VAL, VER 프로세스 영역을 기반으로 관리 활동을 수행하게 된다. 결함 발견 및 제거 활동이 성숙도 수준 2 기반에서는 테스팅만을 수행하였지만, 성숙도 수준 3기반에서는 공식적인 검토와 테스팅을 통해 이루어진다. 그리고 결함을 관리하는 수준은 조직 차원에서 이루어져야 하며, 결함 관리 활동을 하며 문제점이 드러난 프로세스에 대해 개선 활동이 일어난다. 따라서 결함이 어느 단계에서 유입되었는지에 대한 분석이 필요하다. 조직 차원의 결함 분석은 조직 전체를 위한 체크리스트, 프로세스, 또는 교육 훈련 등의 확장으로 연결될 수 있다[2].

CMMI 성숙도 수준 4, 5는 정량적 데이터 관리를 통한 프로세스 개선이 가장 큰 흐름이다. 특히 성숙도 수준 5의 CAR 프로세스 영역에서는 기존의 데이터들의 분석을 통해 결함 예방을 구현하는 것을 목표로 한다.

본 논문에서는 결함 관리의 자동화 및 체계화를 위하여 CMMI 기반의 결함 관리 시스템 Winner를 구상하였다.

5. Winner: CMMI 기반의 결함 관리 시스템

Winner는 기존의 결함 관리 시스템들의 문제점을 해결하고자 CMMI의 단계적 표현에 기반한 결함 관리 프로세스를 지원하도록 구상하였다. 조직에서 만든 제품의 개발 중 혹은 인수 후의 결함 데이터를 수집, 추적, 분석할 수 있도록 하였다.

5.1 Winner 개발 환경 및 시스템 구성

본 시스템의 개발 환경 및 시스템 구성은 아래 그림 7과 같다.

5.2 Winner의 주요 기능

Winner는 기존의 결함 관리 시스템과 기능 면에서는 유사하나, 운영 측면에서 CMMI 의 성숙도 수준의 특성에 따라 결함 관리를 지원할 수 있도록 설계하였다. Winner가 제공하는 기능은 다음과 같다.

1. CMMI 성숙도 수준에 따른 결함 데이터 관리
2. 조직에 맞는 새로운 결함 유형 정의 가능
3. 결함의 상태, 현재 처리 담당자(역할) 할당

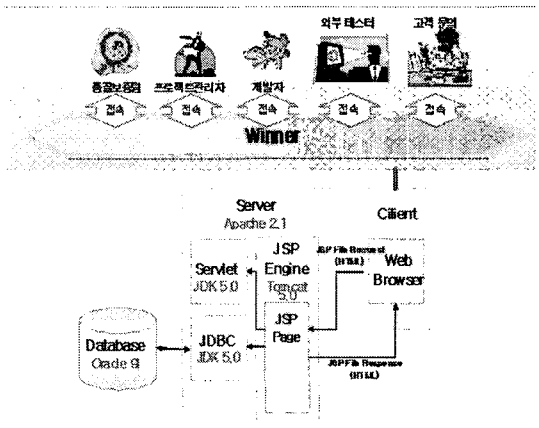

그림 7. Winner 개발 환경 및 시스템 구성도

4. 결함 유형 분석, 결함의 유입 단계 분석
5. 결함의 발견 단계 정보 관리
6. 단계별, Impact 별 결함 개수 및 공수에 대한 통계 그래프 제공
7. 사용자 역할에 따른 권한 관리 기능
8. 사용자에게 메일 전송

5.3 Winner를통한 성숙도 수준에 따른 결함 관리 활동

CMMI 성숙도 수준 2에 기반하여 Winner를 이용한 결함 관리 활동 시 조직은 그림 8의 순서에 따라 작업하게 된다.

성숙도 수준 2에서는 테스팅을 수행하여 결함을 발견하고 제거한다. 결함을 제거 후 성숙도 수준 2의 MA 프로세스 영역에 기반한 측정 데이터를 수집하고 분석을 위해 다음의 내용을 결함 리포트로써 작성하여 Winner에 입력한다.

성숙도 수준 2기반의 결함 리포트는 표 2에 정리된 내용과 같다.

성숙도 수준 2 기반의 결함 리포트는 결함 보고자 및 담당자가Winner에 입력하는 데이터를 말하며, 분석 데이터는 프로젝트 관리자 및 경영진이 볼 수 있는 데이터를 의미한다.

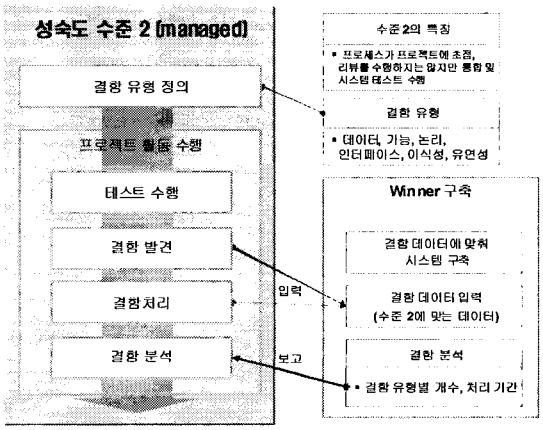

그림 8. 성숙도 수준 2 기반의 결함 관리 활동

표 2. 성숙도 수준 2기반의 결함 리포트

번호	데이터	데이터 설명	입력/분석
1	Product ID	결함이 발견된 프로젝트 또는 서비스	입력
2	Product Component ID	결함이 발견된 프로젝트 밑에 하위 레벨	입력
3	프로젝트 상태	프로젝트가 개발 중/인도 후 인지를 표시	입력
4	결함 유형	결함 분류	입력
5	요약 기술	결함이 어떠한 과정을 거쳐 일어났는지 자세한 설명	입력
6	처리 상태	결함 처리가 진행되는 상태	입력
7	발견일	결함을발견한 날짜	입력
8	처리일	결함을처리한 날짜	입력
9	결함 제거 단계	요구사항/설계/코딩/테스팅	입력
10	담당자	결함을처리하는 담당자	입력
11	인도 후 결함수	인도 후 발견된 전체 결함 개수	분석
12	결함 유형별 개수	결함 DB에 입력된유형별 결함 수	분석
13	결함 처리기간	발견일부터 처리일까지 공수	분석

CMMI성숙도 수준 3을 기반으로 결함 관리를 적용하려는 조직이Winner를 활용하여 결함 관리 활동 시 그림 9의 순서에 따라 작업하게 된다.

성숙도 수준 3에서는 결함을 발견하는 활동이 VAL, VER 프로세스 영역에 기반하여, 테스팅과 소프트웨어 개발 각 단계 종료 후 수행되는 리뷰로 수행된다. 이를 통해 조직은 각 단계별로 발견된 결함의 유형을 구분할 수 있으며, 이들 간의 상관관계를 분석할 수 있다.

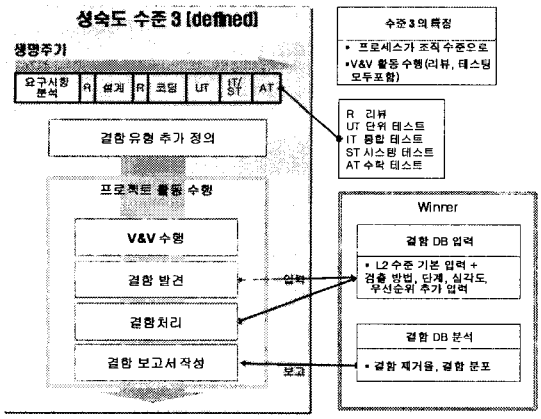

그림 9. 성숙도 수준 3 기반의 결함 관리 활동

Winner를 통해 입력/분석 되는 성숙도 수준 2 기반의 결함 관련 리포트의 추가내용을 추가사항 표 3에 정리하였다.

성숙도 수준 4 기반의 결함 관리는 결함의 발생 단계를 더욱 세분화하여 세분화된 단계마다의 결함 발생을 정량적으로 관리하고, 이상점 발생시 원인의 분석을 수행한다.

표 3. 성숙도 수준 2기반의 결함 리포트 추가내용

번호	데이터	데이터 설명	입력/분석
1	결함 검출 방법	Review or Inspection or Test	입력
2	결함 유입 단계	요구사항/설계/코딩 / 테스팅	입력
3	결함 제거 단계	요구사항/설계/코딩 / 테스팅	입력
4	Impact	결함의영향력 (4단계)	입력
5	결함 주입률	전체 결함수/ 크기(FP, LOC)	분석
6	Impact별 결함수	Impact에 따라 결함 수를 분석	분석
7	검출 방법별 결함수	검출방법에 따라 결함 수 분석	분석
8	결함 분산	단계별에 결함 수	분석

성숙도 수준 5 기반에서는 영향도가 큰 주요 결함에 대해 데이터를 수집하여 근본적인 원인을 제거하여야 한다.

5.4 Winner 입력 화면

CMMI 성숙도 수준에 따라 결함 유형에 맞추어 결함 데이터를 입력받는다.

성숙도 수준 3에 따른 사용자 입력 화면은 그림 11과 같이 성숙도 수준 2에서 입력한 결함 데이터에 확인과 검증 프로세스 영역을 통해 도출할 수 있는 결함 검출 방법, Impact, 결함 유입 단계, 결함 처리 단계의 결함 상세 정보를 추가로 입력한다.

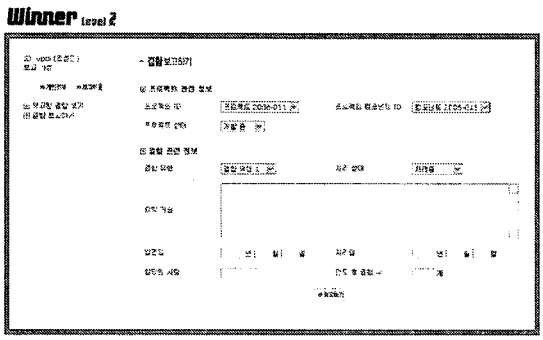

그림 10. 성숙도 수준 2에 따른 사용자 입력 화면

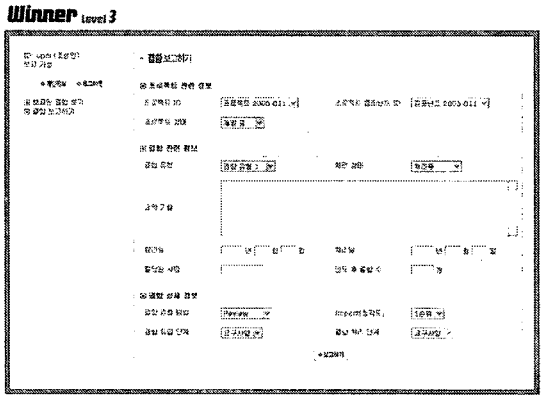

그림 11. 성숙도 수준 3에 따른 사용자 입력 화면

5.5 Winner 주요 분석 정보

그림 12는 주요 분석 정보 중 결함 유형별 통계 그래프에 대한 화면이다. 이를 통해 프로젝트 관리자는 해당 프로젝트에 어느 유형의 결함이 많이 발생했는지 파악이 가능하다. 프로젝트 관리자는 이러한 통계 자료를 근거로 이후 유사 프로젝트 진행 시 이러한 유형의 결함에 대비, 개발자에게 부족한 기술 교육이나, 공수의 조정, 장비 추가 등 사전에 프로젝트의 위험을 낮출 수 있다.

보고된 결함은 그림 13과 같은 화면을 통해서 볼 수 있다.

6. 결론 및 향후 연구 방향

본 논문에서 제안한 CMMI 기반 결함 관리 프로세스 및 결함 관리 시스템을 통해 프로젝트 관리자는 다음과 같은 효과를 볼 수 있다.

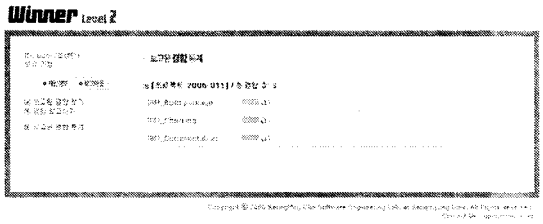

그림 12. 결함 유형 분석 화면

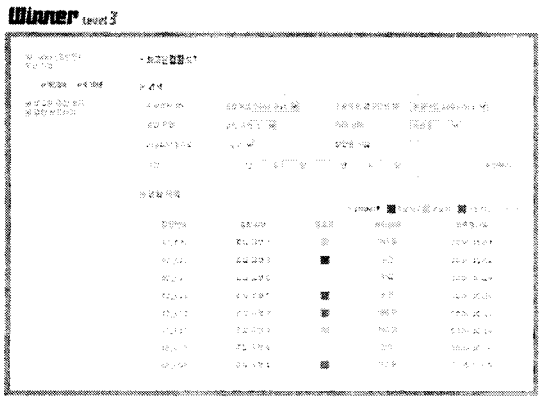

그림 13. 결함 목록 및 검색 화면

1. 결함 데이터를 통해 개발자 개인 별 취약점을 파악하고 이를 프로젝트 관리 및 계획에 반영할 수 있다.

2. Impact가 큰 결함의 재발을 막기 위한 예방 차원의 조치를 취할 수 있다.

3. 프로젝트 관리자는 결함을 제거하는 방법이 어떤 것이 효율적인지에 대한 정보를 정량적으로 파악할 수 있다.

조직 차원에서는 다음과 같은 효과를 볼 수 있다.

1. 결함 관리를 처음 시작하는 조직과 CMMI를 성숙도 수준 2 또는 3의 조직이 CMMI를 기반으로 하는 결함 관리 프로세스 및 시스템을 활용하여 결함 관리를 체계적으로 할 수 있다.

2. 조직이 CMMI 성숙도 수준별로 결함 데이터 수집이 가능하다.

3. 조직 차원에서는 어느 단계, 어느 프로세스에 결함이 많이 유입되는지 파악함으로써 조직의 표준 프로세스 개선 정보에 도움을 줄 수 있다.

본 결함 관리 프로세스는 결함 관리를 처음 시작하는 조직이 결함 데이터를 CMMI 성숙도 수준을 기반으로 관리할 수 있도록 하였으며, 각 수준마다 필요한 결함 관리에 대한 지침도 제시하였다. 이를 통해 처음 결함 관리를 시작하는 조직은 물론이며CMMI 기반의 조직이 품질을 정량적으로 관리할 수 있는 토대를 마련하였다.

향후 연구에는 결함 발생 단계별 유형을 구분하는 방법과 각 단계의 결함 유형간 상관관계를 규명하는 연구가 필요하다. 또한 각 결함 유형별 원인 분석의 연구가 이루어져야 한다.

참 고 문 헌

[1] Brehmer, C., Carl, J. R., "Incorporating IEEEstd 1044 into Your Anomaly Tracking process,"

Crosstalk, No. 40, pp. 9-16, Jan. 1993.
[2] Mary Beth Chrissis, et al., "CMMI guidelinesfor process integration and productimprovement," Addison-Wesley Professional, 2003.
[3] 이민재, 박남직, "CMMI의 이해," ㈜피어슨에 듀케이션코리아, 2006.
[4] Nancy S. Eickelmann and Debra J. Richardson, "A Defect Prevention Approach to Architecture-Based Testing," Pro. COMPSAC '97 - 21st International ComputerSoftware and Applications Conference, IEEE, 1997.
[5] Mitch Allen, "Bug Tracking Basics - A beginner's guide to reporting and tracking defects," STQE, pp. 20-24, May/June 2002.
[6] http://linas.org/linux/pm.html.
[7] Wohlin, Runeson, Defect Content Estimations from Re-view Data, Proceedings InternationalConference on Software Engineering ICSE, pp. 400-409, 1998.
[8] Bruce Durbin, "Q/A Defect TrackingProcess," www.stickyminds.com, 2001.
[9] Pankaj Jalote, 송태국, 이비즈온 SEOG 공역, 구현 사례를 통한 CMM 이해, PersonEducation Korea, 2002.
[10] J.H. van Moll, J. C. Jacobs, B. Freimut, J.J.M. Trienekens, "The Importance of Life Cycle Modeling to Defect Detection and Provention," Pro. 10thInternational Workshop on Software Technology and Engineering Practice (STEP '02), IEEE, 2002.
[11] http://testingfaqs.org/t-track.html.
[12] http://www.mantisbt.org/.
[13] Matthew P. Barnson, "The Bugzilla Guide"
[14] N. E. Fenton and S.L Pfleeger. Software Metrics, a Rigourous and Practica Approach, second edition. International Thomson Computer Press, 1996.
[15] http://www.bugzilla.org/docs216/html/, April 2003.

한 동 준

• 2006년~현재 상명대학교 일반대학원 컴퓨터과학과 석사과정
• 관심분야 : CMM, CMMI, Management

조 성 민

• 2007년 상명대학교 대학원(공학석사)
• 2007년~현재 Teradata
• 관심분야 : SPI, Project Management, Methodology, 6 sigma, CMM, CMMI

이 용 섭

- 2006년~현재 상명대학교 일반대학원 컴퓨터과학과 석사 과정
- 관심분야 : CMMI, Software Quality, Inspection, Project Management, Embedded System

한 혁 수

- 1987년~1992년 미국 South Florida 주립대 컴퓨터공학과 공학박사
- 1985년~1987년 서울대학교 계산통계학과 컴퓨터과학전공 석사
- 1981년~1985년 서울대학교 계산통계학과 컴퓨터과학전공 학사
- 1993년~현재 상명대학교 소프트웨어학부 교수
- 관심분야 : CMM, CMMI, 6 sigma, Process