Embedded SW 의 품질 측정 프로세스 관리 방법에 관한 연구

박 복 남
삼성전자㈜ 디지털미디어연구소

Quality Measurement Process Management Using Defect Data of Embedded SW

Bok-Nam Park
Digital Media R&D Center
Samsung Electronics Co., LTD

요 약

Embedded 소프트웨어의 품질 측정 프로세스 관리는 Embedded 시스템의 적시성과 품질 만족을 위해서도 필요하다. 그러나, Embedded 소프트웨어의 결함에 대하여 사전 분석하거나 예측 없이 개발 프로세스 상에서 결함을 관리하는 것은 위험이 따른다. 본 연구에서는 Embedded 소프트웨어에서 품질 측정 프로세스 관리를 위해 소프트웨어의 정량적 속성 중에 가장 중요한 요소 중에 하나인 결함을 중심으로 본 연구가 진행되었다. Embedded 소프트웨어에 가장 적합한 프로세스를 정의하고 개선하고자 하는 과정에서, 프로세스 관리를 효과적으로 수행하기 위해 Embedded 소프트웨어의 특성과 결함 특성을 이해하고, 이를 근간으로 결함 속성을 정의하고 결함을 통한 품질 측정 프로세스 관리를 할 수 있도록, 결함 데이터를 이용하여 프로세스를 관리하는데 기여하고자 한다. 따라서, 본 연구에서는 결함 데이터 분석을 위해 필요한 속성을 파악하고, 테스트 단계를 중심으로 결함 데이터의 활용과 결함 데이터를 이용한 프로세스 관리 방법을 제안하여, 이를 통해 Embedded 소프트웨어 프로세스를 관리하는 분들에게 효과적인 활용이 될 수 있도록 한다.

The time to market and productivity of embedded system needs a quality measurement process management of embedded software. But, defect management without preemptive analysis or prediction is not useful for quality measurement process management. This subject is focused on the defect that is one of the most important attributes of software measure in the process. Defining of defect attribute and quality measurement process management is according to understanding of embedded sw characteristics and defect data. So, this study contributes to propose the good method of the quantitative based on defect management in the test phase of sw lifecycle.

Key Word : Embedded SW, Software Quality Measurement, Software Process Improvement, CMM, Defect Management, Test

1. 서론

Embedded 란 하드웨어나 소프트웨어가 다른 하드웨어나 소프트웨어의 일부로 내포되어 있는 내장형을 의미한다. 최근 Embedded 소프트웨어 시장이 빠르게 성장하고 있는데 Gartner Dataquest 에 따르면 2000 년 전세계 Embedded 소프트웨어 시장 규모가 8 억 6,600 만 달러에 이르러 1999 년의 6 억 2,300 만 달러 대비 37%의 성장률을 기록한 것으로 나타났다.

또한, Embedded 소프트웨어가 점차 복잡해 지고 소프트웨어 크기(KSLOC) 면에서도 점차 증대되고 있는 추세와 더불어, Embedded 소프트웨어가 시스템에서 중요한 기능을 담당하고 시스템 품질의 주요한 요인으로 인식하게 됨에 따라, 실제로 Embedded 소프트웨어를 개발하는 프로세스가 요즈음 주요한 이슈(Issue)로 등장하고 있다.

Embedded 소프트웨어 개발 담당자나, 품질보증

담당자는 개발 생산성과 품질 향상을 위하여 개발 능력 면에서 자질이 검증된 개발자를 선발하거나, 교육을 통해 이를 극복하려고 하고 있는 한편, 프로세스의 국제 표준인 ISO12207, 프로세스의 수준을 평가하고 개선하는 모델을 제공하는 SPICE(Software Process Improvement and dEtermination) 및 CMM(Capability Maturity Model)을 도입하려는 시도를 함께 하고 있는 것이 요즈음의 현실이다.

즉, 프로세스가 개발 생산성과 품질 향상의 주요한 결정 요소로 인식하면서, ISO12207 의 프로세스를 근간으로 SPICE, CMM 등의 개선 모델을 도입하여 추진하고 있지만 기업의 일반사무용 application 소프트웨어의 개발 프로세스를 답습하거나, 일부 조정(Tailoring)하여 적용하는 경우가 적지 않다.

SEI(Software Engineering Institute)의 CMM(Capability Maturity Model)에 따르면 조직의 성숙도가 높아지려면 프로세스의 정량적 관리가 필요하다[1][2]. 프로세스의 정량적 관리에는 공수/비용, 일정, 주요 컴퓨터 자원 사용량 등의 측정치가 사용될 수 있으나[3], 개발 프로세스의 정량적 관리에는 결함 데이터가 중요한 역할을 차지한다[4].

소프트웨어 결함은 시스템의 품질을 결정하는 주요한 관리 요소일 뿐 아니라 소프트웨어 프로세스를 측정 및 관리하는 수단으로 활용될 수 있다. 소프트웨어 제품과 프로세스를 측정하는 이유는 일정, 비용, 제품의 품질을 효과적으로 관리하기 위한 데이터를 수집하는 것이다. 특히 소프트웨어 제품에서 발견되는 문제와 결함의 측정은 시정 조치 여부 결정, 소프트웨어 개발 프로세스의 측정 및 개선, 잔존 결함 수 및 실패율의 예측에 매우 중요하다[5][6].

본 연구에서는 Embedded 소프트웨어에 가장 적합한 프로세스를 정의하고 개선하고자 하는 과정에서 프로세스 관리를 효과적으로 수행하기 위해 Embedded 소프트웨어의 특성과 결함 특성을 이해하고 이를 근간으로 결함 속성을 정의하고 결함을 통한 품질 측정 프로세스 관리를 할 수 있도록 결함 데이터를 이용하여 프로세스를 관리하는데 기여하고자 한다. 따라서, 결함 데이터 분석을 위해 필요한 속성을 파악하고, 테스트 단계를 중심으로 결함 데이터의 활용과 결함 데이터를 이용한 프로세스 관리 방법을 제안하고자 한다.

2. Embedded 시스템의 경향

Embedded 시스템에 따른 소프트웨어의 특성은 다음과 같다[7].

2.1 Embedded 시스템의 경향

Embedded 시스템이란 특정한 목적을 수행하도록 만든 컴퓨터 하드웨어와 소프트웨어의 결합체이다. 즉, Embedded 시스템에서 컴퓨터는 특정 목적을 위해 설계된 시스템의 한 구성요소이다. 이러한 Embedded 시스템이 반도체 기술과 통신 기술의 급속한 발전과 더불어 인터넷 보급, 멀티미디어라는 정보 매체를 처리하기 위해 Embedded 시스템의 설계 복잡성은 빠르게 증가해 왔다.

이런 제반 기술의 급속한 발전은 Embedded 시스템의 소형화/경량화, 고성능화, 다기능화 및 다양화를 가능하게 하였다. 현재에 와서는 엄청난 양의 특화된 하드웨어와 아주 복잡한 소프트웨어가 효과적으로 실행되도록 설계된 고성능 프로세스가 하나의 칩 안에 집적되게 하는 SoC(System- on-Chip) 개발이 일반화되어 가고 있다[8].

2.2 Embedded 소프트웨어의 경향

과거에 하드웨어와 소프트웨어가 독립적으로 개발된 후, 설계의 마지막 단계에서 결합되는 설계 방법으로는 도저히 시장의 고성능 및 빠른 생산 시간(fast time to market) 요구 조건을 만족시킬 수 없으며, 하드웨어와 소프트웨어 간의 끊임없는 상호 연관을 통하여 동시에 개발/검증되는 방법이 Embedded 소프트웨어 개발 프로세스에 있어 매우 중요하다[9].

3. Embedded 소프트웨어의 특성

3.1 Embedded 소프트웨어의 일반 특성

Embedded 시스템이 너무나 다양해서 특징을 일일이 언급한다는 것이 어렵듯이, Embedded 소프트웨어의 특성을 나열하는 것이 힘들지만 앞에서 열거한 경향에 의거하여 몇 가지 특성을 요약하여, 이를 표로 정리하면 <표 3-1>과 같다[10][18].

• 목적의 한정성

Embedded 시스템이 어떤 특정 목적을 위해 설계될 때, 수행되는 기능이 거의 고정적이기 때문에, 범용 시스템의 설계 시 만큼, 소프트웨어만을 최적화시켜 얻을 수 있는 성능 향상이 크지 않은 경우가 많다.

• 실시간 처리성

범용 시스템의 처리는 일반적으로 주어진 자원을 최대한 효과적으로 활용하여, 되도록 빠르게 수행하는 것을 목적으로 한다. 이에 반해 Embedded 시스템의 경우는 과제(Task)의 처리 기한이 주어지는 실시간 처리가 많다.

• 욕구의 다양성

우리의 생활 속에서 쉽게 발견되는 몇 가지 Embedded 시스템의 예를 보아도, Embedded 시스템이 얼마나 우리의 실생활과 밀접하게 연관되어 있는지 알 수 있다. 즉, Embedded 시스템의 성능 평가는 최종 제품의 성능에 의해 평가되지, 그 안에 쓰이는 하드웨어와 소프트웨어의 성능에 평가되지는 않는다.

• 강한 내구성

많은 Embedded 시스템은 고온이나 다습한 환경, 또는 충격이 가해지거나, 일부 기능에 이상이 있어도 기본 기능은 계속 동작하도록 요구되는 경우가 많다.

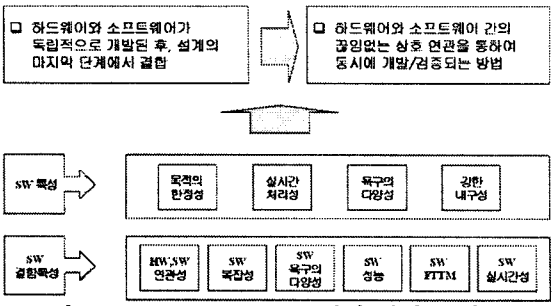
구분	소프트웨어 개발 특성	비고
목적의 한정성	SW와 HW 동시 개발 -기능의 HW,SW 구현 여부 먼저 고려 -HW와 SW 연관성 중점 -제한된 메모리 용량	
실시간 처리성	HW,SW Sync. 중점 설계 -작업의 우선순위 설정	

	-작업 실행 Deadline 설정 -Event Reactive 설계	
욕구의 다양성	소비자의 다양한 욕구 -고객 중점 -요구사항 검증 중점	
강한 내구성	환경 다양화 -다양한 환경의 신뢰성 -오류/예외 중점 검증	

<표 3-1> Embedded 소프트웨어의 특성

3.2 Embedded 소프트웨어 결함 특성

Embedded 시스템에 따른 소프트웨어 결함의 특성은 다음과 같으며, 이를 그림으로 정리하면 <그림 3-1>과 같으며, 표로 정리하면 <표 3-2>와 같다 [11].



<그림 3-1> Embedded 소프트웨어 결함 특성도

3.2.1 HW, SW 연관성

과거의 Embedded 소프트웨어의 설계는 하드웨어와 소프트웨어의 독립적인 개발팀에 의해 먼저 하드웨어가 설계되고, 소프트웨어는 이미 정해진 프로세서에 수행된다는 가정 하에 독립적으로 개발되는 형태로 많이 이루어져 왔다. 하지만, 이제는 그런 방법에 의한 설계는 경쟁력이 없다. 즉, 앞으로의 Embedded 시스템의 설계는 하드웨어와 소프트웨어가 동시에 개발되면서, 가장 최적의 하드웨어 구조와 또 그 하드웨어를 최적으로 이용할 수 있는 소프트웨어 개발이 동시에 밀접한 상호작용을 갖으며 이루어져야 한다.

3.2.2 SW 복잡성

과거에는 특정 목적을 갖고 있는 Embedded 시스템이 한 두가지의 목적만을 위해서 설계되었기 때문에 대부분 간단했다. 실제로 대부분의 소프트웨어는 하나의 기기의 수명이 다하는 동안 항상 일정한 기능만을 실행하기 때문에 ROM에 들어가 있는 경우가 일반적이었다. 현재에 들어 메모리 가격의 인하와 Flash 메모리의 폭넓은 보급으로 소프트웨어의 활용이 더 용이해지면서 소프트웨어의 장점이라 할 수 있는 유연성과 기능 확장의 용이성이 가능한 점까지 더해져 점점 더 비중이 증가하고 있다.

3.2.3 SW 욕구 다양성

Embedded 시스템이 실 생활의 제품에서 다양하게 선보이면서 불특정 다수의 고객, 고객 욕구의 다양성, One-to-One 마케팅의 제품 수용 등으로 소프트웨어도 이러한 욕구의 다양성에 많은 영향을 받게 되었다. 더불어, 최근의 휴대폰이나 멀티미디어 처리 기기는 소비자의 욕구가 자주 바뀌면서 유연한 기능의 개선이 가능해야 하는 요구사항까지 존재하게 되었다.

3.2.4 SW 성능

과거의 Embedded 시스템이 일반적으로 보드(Board)에 시스템이 구현되는 형태였으나 최근 들어 급속도로 발전된 반도체 기술은 엄청난 양의 복잡한 하드웨어를 하나의 칩 안에 넣을 수 있는 것이 가능해 졌다. 이는 초소형/초경량 시스템의 일반화를 의미하는 것으로 다방면에서 큰 파급효과를 주는 중요한 경향이라고 할 수 있다.

3.2.5 SW FTTM(Fast-Time-To-Market)

Embedded 시스템은 범용 컴퓨터의 설계에 비해 대량 생산을 하면서도 치열한 경쟁 속에서 다양한 소비자의 욕구를 만족시키기 위해서는 빠른 시간 내에 경쟁력 있는 제품을 출시해야 하는 부담이 있다. 이를 위해서는 빠른 설계가 필수적인데, Embedded 시스템이 나날이 복잡해짐에 따라 빠른 시간 안에 모든 설계를 처음부터 시작해서 끝낸다는 것은 현실적으로 거의 불가능하다.

3.2.6 SW 실시간성

Embedded 시스템은 실시간 처리를 요구하는 경우가 많다. 이와 같이 각 작업들이 처리해야 하는 최소 처리 속도나 처리기한이 주어져 있을 때, 작업 처리의 기준이 모든 작업의 실행 제약시간에 어김없이 처리하도록 요청된다.

단계	구분	결함특성	내용
계획	목적의 한정성	3.2.1	WBS-Task 충분성 -HW,SW 연동작업 -HW,SW 동시작업
	목적의 한정성	3.2.5	SDLC 적정성 -프로세스적정성
	실시간 처리성	3.2.4	성능 계획성
	실시간 처리성	3.2.6	실시간처리 계획성
	욕구의 다양성	3.2.2	기능 구현 관리성 -일정/크기관리성 -변경관리성
	욕구의 다양성	3.2.3	고객 명확성 -요구사항 충분성 -요구사항 명확성
	강한 내구성	3.2.1	HW,SW 연동작업
분석	목적의 한정성	3.2.1	WBS-Task 충분성 -HW,SW 통합성
	목적의 한정성	3.2.5	변경관리 적정성
	실시간 처리성	3.2.4	성능 분석성
	실시간 처리성	3.2.6	실시간 처리 설계성
	욕구의 다양성	3.2.2	기능분석 충분성
	욕구의 다양성	3.2.3	변경관리 적정성
	강한 내구성	3.2.1	HW,SW 통합성
설계	목적의 한정성	3.2.1	WBS-Task 충분성 -HW,SW 통합성
	목적의 한정성	3.2.5	변경관리 적정성
	실시간 처리성	3.2.4	성능 설계성
	욕구의 다양성	3.2.2	기능설계 충분성
	욕구의 다양성	3.2.3	고객 명확성 -Testcase 충분성 -Testcase 명확성 변경관리 적정성
	강한 내구성	3.2.1	HW,SW 통합성
구현	목적의 한정성	3.2.1	WBS-Task 충분성 -HW,SW 통합성
	목적의 한정성	3.2.5	변경관리 적정성

	실시간 처리성	3.2.4	성능 구현성
	실시간 처리성	3.2.6	실시간 처리 구현성
	욕구의 다양성	3.2.2	기능 구현 충분성
	욕구의 다양성	3.2.3	변경관리 적정성
	강한 내구성	3.2.1	HW,SW 통합성

<표 3-2> Embedded 소프트웨어 결함 특성 및 내용

4. 정량적 관리 속성으로서의 결함 분석

4.1 소프트웨어 결함

결함은 프로그램 상에서의 문법에러(syntax error), 부정확한 프로그램 문장(incorrect program statement), 철자상의 에러 등을 의미한다[12]. 결함은 소프트웨어 개발에 있어서 설계 및 코딩단계에서 주로 나타나며, 요구사항분석단계 및 배포단계에서도 발견된다. 특히, 결함은 소프트웨어 사용자에게 미치는 영향뿐만 아니라 결함을 발견하고 제거하는 비용 및 일정의 측면에서도 매우 중요한 개념이다[13].

William은 소프트웨어 결함을 소프트웨어 산출물이나 프로세스의 결점이나 불완전성이라고 정의하여 제품 자체 뿐 아니라 프로세스에까지 결함의 정의를 확장하였다[14]. 본 연구에서는 William이 사용한 결함의 정의를 사용하기로 한다.

4.2 결함의 속성(Attributes)

문제나 결함은 다양한 방법으로 보고되고 기록될 수 있지만 각각의 유사한 특성으로 분류가 가능하다. David은 유입 단계, 발견 단계, Major/Minor, 우선순위와 심각도, 에러 형태, 발견일/해결일, 관련 적용 분야 등 7가지로 결함을 분류하였으며[4], William은 누가, 무엇을, 왜, 언제, 어디서, 어떻게의 여섯 가지 질문에 근거한 소프트웨어 측정 프레임웍을 이용해 다음과 같은 일반적인 문제와 결함의 속성을 정의하였다[14][15]. 이를 토대로 테스트 단계를 중심으로 실제 적용 항목 비교를 <표 4-1>과 같이 하였다[19].

속성	적용
Identification	Sw product,deliverables,etc
Finding Activity	Phase/Activity/Task, inspections,formal reviews,testing,customer support,undetermined
Finding Mode	Static, dynamic, undetermined
Criticality	Critical,major,minor,cosmetic
Problem Status	Open,recognized,evaluated,resolved,closed
Problem Type	Sw defects, other problems, undetermined*(Issue,Next Ver., Un-reappear, etc)*
Uniqueness	Original, Duplicate, value not identified
Urgency	Most urgent, average, ...
Environment	OS, Platform
Timing	Time for detect,resolved,etc
Originator	Tester,QA,etc
Defects Found In	Module
Changes Made To	Module
Related Changes	Related Module
Projected Availability	Expected Date
Released/Shipped	Configuration status(version)
Applied	Version
Approved By	Approval
Accepted By	Submittee

<표 4-1> 결함속성 대비 적용 비교표

4.3 결함의 관리 수준

최근에 프로세스를 개선하고 수준을 측정하기 위한 표준으로 CMM을 적용하고 있다. 이를 근간으로 프로세스 수준별 결함을 정의하여 <표 4-2>로 요약하였다[16].

레벨	관리 척도
레벨 2	결함 또는 에러/KSLOC(실제 또는 예측된 KSLOC) [코딩 단계]
	예상 결함/KSLOC [배포 단계]
	(처리된 PTR/전체 PTR)×100(%)
	PTR/KSLOC [통합시험 단계에서]
	PTR/KSLOC [시스템 시험 단계에서]
레벨 3	결함 또는 에러/KSLOC(실제 또는 예측된 KSLOC) [PDR, DDR 단계]
	결함 수/크기
	결함 또는 에러/KSLOC(실제 또는 예측된 KSLOC) [PDR, DDR 단계]
	(완결된 SAI/전체 SAI)×100(%) [코드 검사단계에서 발생된 SAI 추적]
	결함이나 에러/KSLOC(실제 또는 예측된 KSLOC) [코딩 단계]
레벨 4	계획 대비 실제 결함 또는 에러수(결함 또는 에러/KSLOC)[PDR, DDR 단계]
	계획 대비 실제 결함 또는 에러수(결함 또는 에러/KSLOC)[코딩 단계]
	결함 또는 에러/KSLOC (실제 또는 예측된 KSLOC) [PDR, DDR 단계]
	단계별 기능성 시험의 시험범위는 측정되고 기록된다
	검토 데이터(review data) 분석 [PDR, DDR, 코딩 단계]

<표 4-2> CMM 레벨 대비 결함 관리 수준

<용어설명>

PTR(Program Trouble Report), DDR(Detailed Design Review), SAI(Software Action Item), PDR(Preliminary Design Review), KSLOC(Thousand Source Lines of Code)

4.4 SW의 가시성 제고를 위한 정량적 관리

Embedded 소프트웨어의 개발 프로세스를 효과적으로 관리하기 위해서는 정량적으로 프로세스를 측정할 수 있어야 한다. 측정에 근거한 소프트웨어 프로세스 관리는 개발 프로세스 산출물들이 정량적 목표를 달성할 수 있도록 유도하고, 점진적으로 목표가 달성된 정도를 평가하는 것을 의미한다[17]. 즉, 개발 프로젝트와 프로세스에 대한 가시성(Visibility)을 높이기 위해서는 프로세스의 정량적 관리를 통해 제품의 크기, 비용, 일정, 품질 등에 대한 이해 가능한 측정이 필요하다.

4.5 결함 측정 요건과 결함 데이터 활용 방법

baumert는 소프트웨어 측정 관점에서 CMM 각 단계별 요건을 비교하고 각 단계에서 사용되는 측정치에 대한 세부 요건을 설명하였다[2]. 본 절에서는 baumert의 CMM 각 단계별 측정 요건에 대한 설명을 소개하고, 각 단계에서 문제 보고 데이터의 활용 방법에 대해 <표 4-3>에서 요약 설명한다.

단계	추세분석	관계 분석
2 단계	소프트웨어 관리자가 소프트웨어 제품의 품질,	

	신뢰성, 테스트의 효과성을 파악함	
	1.결함 보고의 수, 유형, 심각성 2.결함 보고의 밀도(단위 크기당 문제 보고 수) 3.결함 보고 비율	통과된 테스트 케이스 수와 결함 보고 수와의 관계
3 단계	1.소프트웨어 관리자가 소프트웨어 제품의 품질, 신뢰성, 테스트의 효과성을 파악함 2.소프트웨어 공학 그룹에게 개발 프로세스에 대한 정보를 제공	
	1.문제 보고의 수, 유형, 심각성 2.문제 보고의 밀도(단위 크기당 문제 보고 수) 3.문제 보고 비율 4.문제 보고서가 작성되는 비율 5.소프트웨어 컴포넌트 별 결함 수	통과된 테스트 케이스 수와 문제 보고 수와의 관계
4 단계	제품 및 프로세스 수행에 대한 정량적인 관리 범위를 수립하기 위해 일관되고 잘 정의된 측정치를 사용	
	3 단계 결함 데이터에 문제의 원인, 테스트/개발/구현의 효율성에 대한 추세 분석이 추가	-
5 단계	프로세스 효과성에 대한 정량적인 증거를 사용	
	결함(문제 보고) 데이터는 4 단계와 동일	-

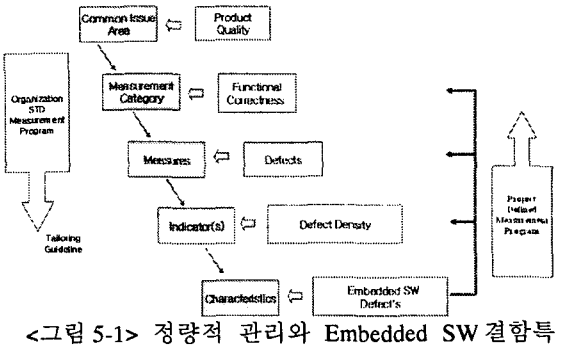
<표 4-3>단계별 결함 분석

5. 결함 데이터를 이용한 품질 측정 프로세스 관리

아직까지 품질 측정 프로세스 관리에 대한 사례는 많지 않으며 이와 관련된 검증된 방법도 찾아보기 힘들다[2]. 결함 데이터는 프로세스를 정량적으로 관리할 수 있는 중요한 수단 중 하나이며 조직의 성숙도를 높이는데 중요한 역할을 한다[4]. 본 장에서는 테스트 단계에서 결함 데이터를 이용해 프로세스를 관리하는 방법에 대해 제안하고자 한다.

5.1 정량적 관리와 Embedded SW 결함 특성 관계 모델

전사 혹은 프로젝트의 목적을 달성하기 위한 품질 측정 프로세스 관리는 먼저, 각 공통이슈영역(Common issue area), 측정 카테고리(Measurement category), Measures, Indicator(s)를 정의하는 절차와 함께 Embedded SW의 결함 특성을 프로젝트에 적용하기 위한 모델을 제시한다[21]. 이를 요약하여 나타낸 그림은 <그림 5-1>과 같다.



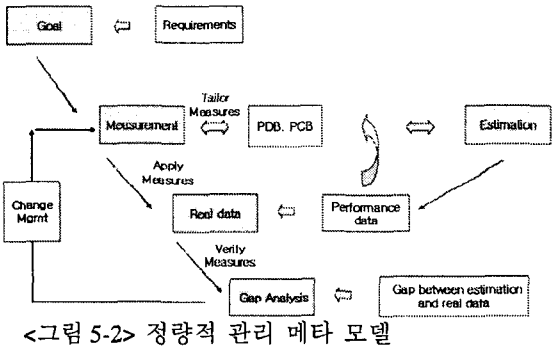
<그림 5-1> 정량적 관리와 Embedded SW 결함특성 관계모델

5.2 정량적 관리와 결함 관리 메타 모델

프로세스에서 정량적 관리의 기본 모델인 GQM(Goal-Questions-Metirc) 모델을 기본으로하여 정량적 관리와 결함 관리 메타 모델을 소개한다[19].

5.2.1 정량적 관리 메타 모델

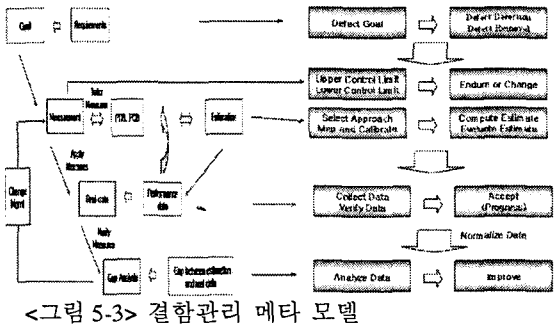
요구사항을 근거로 하는 목표 수립, 측정 정의, 실제 측정, 차이 분석을 통한 프로세스 관리 모델이다. 이를 요약하여 나타낸 그림은 <그림 5-2>와 같다.



<그림 5-2> 정량적 관리 메타 모델

5.2.2 결함 관리 메타 모델

정량적 관리 메타 모델을 근간으로 실제 적용가능한 결함관리 절차를 분석하여, 이를 근간으로 프로세스를 정량적으로 관리하고자 하는 절차를 정의하였다. 이를 요약하여 나타낸 그림은 <그림 5-3>과 같다.



<그림 5-3> 결함관리 메타 모델

5.3 정량적 관리 목표 수립

단계별 정량적 관리 목표를 수립한다. 정량적 관리 목표는 각 테스트 단계별로 결함심각도, 결함우

선순위에 대하여 가중치를 정하고, 이를 기준으로 결함제거율, 기능 완료율에 따라 시작과 종료 기준을 정하도록 한다. <표 5-1>은 정량적 관리 목표의 한 예를 보여 준다.

구분	결함 구분	Defect 분류	가중치	단위 시험	통합시험	시스템 시험	인수 시험	비고
결함	결함 심각도	Critical	10	O	O	O	O	O:결함 없음 X:결함 있음
		Major	7	X	O	O	O	
		Average	5	X	X	O	O	
		Minor	2	X	X	X	O	
	결함 우선 순위	긴급	10	O	O	O	O	
		중요	7	X	O	O	O	
		보통	5	X	X	O	O	
		낮음	2	X	X	X	O	
결함 제거율	결함 제거율			80% 이상	90% 이상	95% 이상	100%	
기능 완료율	기능 완료율	시작 기준		NA	80% 이상	90% 이상	100%	
		종료 기준		80% 이상	90% 이상	100%	100%	

<표 5-1> 정량적 관리 목표

5.4 품질 측정 프로세스 관리 기준

정량적 관리 목표가 수립되었을 때, 각 단계별 통과 기준을 정하여 이의 프로세스를 관리하도록 한다. <표 5-2>는 품질 측정 프로세스 관리 기준의 한 예를 보여 준다.

항목	목표		측정 기준	측정 시기	측정자
프로세스 준수율	규격 품질평가	85 %	품질평가 체점 기준	품질평가 시	SQA
	기능 품질평가	85 %			
	일관성 품질평가	85 %			
표준 준수율	UI 표준 준수율	98%	UI Guideline	구현 이후	SQA
	코딩표준 준수율	85 %	Coding Guide의의 차이	구현 이후	SQA
요구사항 구현율	요구사항 구현율	98%	요구사항 대비 구현율	구현 이후	SQA
창의적 변경 수	창의적 변경수	5-10건	창의적 변경 요청/반영수	구현 이후	SQA
프로세스 개선 건수	프로세스 개선 건수	5-10건	개선 요청 수 개선 반영 수	구현 이후	SQA
결함 조치율	Critical	100%	SW 지표관리 기준	Baseline이후	SQA
	Major	100%			
	Average	100%			
	Minor	99%			

<표 5-2> 품질 측정 프로세스 관리 통과 기준

5.5 정량적 측정치 관리 방법

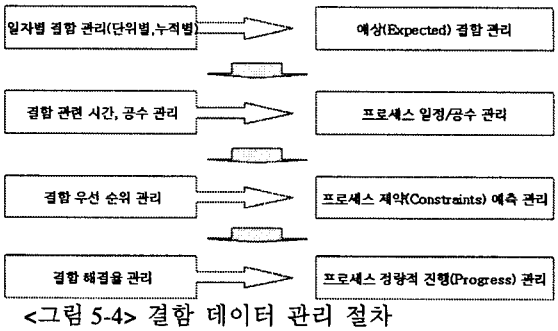
정량적 관리 목표가 수립되고 이를 측정할 관리 지표를 정한다. 이의 관리 지표는 결함, 테스트케이스, 생산성, 창의적변경, 일정에 대하여 정의를 한다. <표 5-3>은 정량적 측정 방안의 한 예를 보여 준다.

구분	관리지표	측정시기	측정방법	작성 주체
결함	구현 이후 테스트 결함조치 율 or 결함밀도	구현 Baseline 이후	테스트 단계별, 구현 단위(CPP) 별 결함 발견 수&공수, 결함 해결/미해결 수 &공수 결함밀도(결함 수/CPP Line)	개발팀->Tester->QAM
	구현 이후 QA 결함조치 율 or 결함밀도	구현 Baseline 이후	테스트 단계별, 산출물 별 결함 발견 수, 결함 해결/미해결 수 결함밀도(결함 수/산출물 Line)	개발팀->QA->QAM
	결함 순위/심각도별 관리	구현 Baseline 이후	테스트 단계별, 산출물 별, 결함 순위/심각도별 결함 발견 수, 결함 해결/미해결 수 결함밀도(결함 수/산출물 Line)	개발팀->QA->QAM
	결함 구분 별 관리	구현 Baseline 이후	테스트 단계별, 산출물 별, 결함 구분별 결함 발견 수, 결함 해결/미해결 수 결함밀도(결함 수/산출물 Line)	개발팀->QA->QAM
테스트 케이스	테스트케이스 수 및 소요시간	각 테스트 단계별	테스트 단계별 신규 작성 테스트케이스 수 및 소요 시간 추가/변경/삭제 테스트케이스 수 및 소요 시간	Tester->QAM
생산성	작업 단위 별 생산성	각 테스트 단계별	테스트 단계별, 구현단위별 테스트케이스 생산성(수/시간) 결함 생산성(발견/해결 수/시간)	개발팀->Tester->QAM
창의적 변경	창의적 기능/비기능 변경 요청 수	각 테스트 단계별	테스트 단계별 창의적 기능/비기능 변경 요청 수	Tester->개발팀->QAM
일정	납기 준수 율	구현 Baseline 포함 이후	각 단계별 완료 단위 수 및 미완료 단위 수 간 백분율	개발팀->Tester->QAM

<표 5-3> 정량적 측정 방안

5.6 결함 데이터 관리 절차

이제 결함을 정량적으로 관리하기 위한 관리 절차를 제안하고자 한다. 하기 제안된 절차는 결함 데이터를 활용하고 분석하기 위한 도구도 함께 제안된다. <그림 5-4>는 결함 데이터 관리 절차를 표현한다.



<그림 5-4> 결함 데이터 관리 절차

5.6.1 예상(Expected) 결함 관리

본 데이터 분석을 통해 향후 예상되는 결함을 관리하고자 하는 것이 목적이다.

테스트 단계를 중심으로 하여 결함 데이터를 분석하여 활용하는 기준은, 먼저 일자별로 결함이 발견되면 일자별 결함 추이 분석을 하면서 누적 결함 추이분석을 한다. 이를 통하여 결함 예상곡선이 도출되고 이를 통해 향후 예측 가능한 미발견 결함수가 파악된다. 물론, 누적 결함 추이에 따른 향후 결함 추이를 분석하는 도구를 사용하는 것이 정확하면서 편리하다.

이와 관련된 도구는 '일자별 결함 추이', '일자별 누적 결함 추이'를 제안한다.

5.6.2 프로세스 일정/공수 관리

본 데이터 분석을 통해 향후 프로젝트 일정 및 공수를 관리하고자 함이 목적이다.

결함과 관련된 주요 시간 및 공수를 파악한다. 예를 들면, 결함 발견 시간 및 공수, 결함 해결 시간 및 공수, 결함 모듈 수정 크기(SLOC) 등을 파악한다. 이를 통하여 향후 예측되는 결함에 대한 평균 결함 관련 시간 및 공수를 파악하고 이를 통해 계획된 관리 내용과 향후 예측되는 내용을 비교하여 차이 분석을 통한 교정(Correction) 관리를 하게 된다. 관련 도구는 일자별 결함 추이, 일자별 누적 결함 추이를 제안한다.

이와 관련된 도구는 '시험 및 결함관리 시간'을 제안한다.

5.6.3 프로세스 제약(Constraints) 예측 관리

본 데이터 분석을 통해 프로세스의 일정/공수/비용과 관련한 제약에 따른 예측 관리를 하고자 함이 목적이다.

결함 우선 순위를 파악하고 이의 데이터를 근거로 하여, 개발 상의 일정/공수/비용 제약사항에 따른 가능한 결함 관리 범위를 파악한다. 결함 우선 순위에 따라 관리 범위 내의 결함 및 향후 차기버전으로 이동되는 결함 등이 분석된다.

이와 관련된 도구는 '결함 우선 순위별 발견 및 해결수'를 제안한다.

5.6.4 프로세스 정량적 진행(Progress) 관리

본 데이터 분석을 통해 프로세스의 진행에 따른 구현 정도를 파악하고 이를 통해 프로세스의 진행 관리를 정량적으로 하고자 함이 목적이다.

결함 발견에 따른 해결 결함율을 파악하고, 각 요구사항에 따른 결함 데이터를 근간으로 하는 구현 분석을 한다. 이를 통해 각 Baseline 혹은 Milestone 별로 구현율과 구현 정도를 파악할 수 있다.

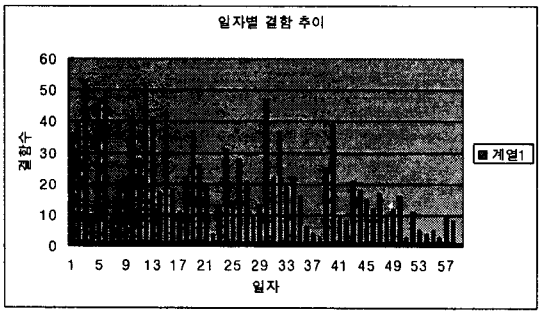
이와 관련된 도구는 '요구기능별 결함 해결율', '요구기능별 구현율'을 제안한다.

5.7 결함 데이터 관리 기준에 따른 활용 방안

결함 데이터의 분석 및 활용은 궁극적으로는 프로세스 개선에 있다. 프로세스 개선을 하는데 있어 결함 데이터를 이용한 관리 방법과 그 활용에 대하여 분석하고자 한다. 그러나 잊지 말아야 할 것은 분석하고자 하는 목적과 방법에 따라 사전에 데이터의 속성 및 수집 방법이 결정되어 있어야 한다는 것이다. 2 장에서 설명한 것처럼 결함 데이터는 여러 가지 속성을 적용할 수 있다. 본 절에서는 제안된 결함 데이터 관리 기준에 따른 요구기능별, 일자별 결함 데이터를 활용한 방법 및 도구를 소개한다.

5.7.1 일자별 결함 추이 분석

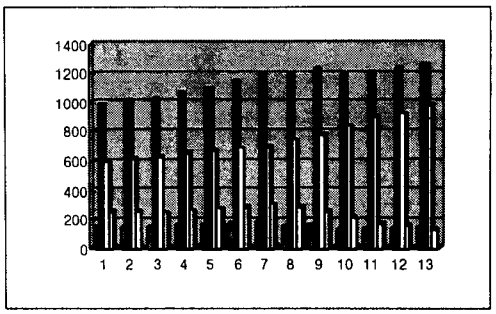
일자별로 발견, 해결, 미해결, 차기버전으로 처리되는 결함에 따른 추이를 관리한다. 일자별로 요구기능별(혹은 전체) 결함 발견수, 해결결함수, 미해결결함수, 차기버전수를 정의하도록 한다. <표 5-4>은 일자별 결함 추이의 한 예를 나타낸다.



<표 5-4> 일자별 결함 추이

5.7.2 일자별 누적 결함 추이 분석

일자별로 누적수치 기준으로 발견, 해결, 미해결, 차기버전으로 처리되는 결함에 따른 추이를 관리한다. 일자별로 누적 수치 기준으로 요구기능별(혹은 전체) 결함 발견수, 해결결함수, 미해결결함수, 차기버전수를 정의하도록 한다. <표 5-5>은 일자별 결함 추이의 한 예를 나타낸다.



<표 5-5> 일자별 누적 결함 추이

<참조설명>
막대구분 : 전체결함, 해결결함, 미해결결함, 차기버전

5.7.3 시험 및 결함 해결, 수정 시간

테스트 활동에 따라 요구 기능별로 시험을 수행하면서 혹은 결함을 해결하면서 소요되는 시간 및 해결할 때 수정되는 소스(Source)의 크기 등에 대한 정량적 데이터를 관리하고자 한다. 이는 시험 및 결함시험 소요 시간, 재 시험소요 시간, 결함 해결 시간, 수정 크기, 평균 결함 해결 시간, 결함 해결 예정 시간을 정의하면서 나타낸다. <표 5-6>는 시험 및 결함 관리 시간의 한 예를 나타낸다.

기능명	시험소요시간	재시험 소요시간	결함 해결시간	수정크기	평균결함 해결시간	결함해결 예정시간
A	1819	803	1684	1831	15.89	2081.59
B	810	150	1040	595	47	2773
C	1622	399	2232	1088	34	4284
계	4251	1352	4956	3514	32.3	9138.59
	70.85Hr	22.5Hr	82.6Hr	3.514	32.3	152.3Hr

<표 5-6> 시험 및 결함 관리 시간

5.7.4 결함 우선순위별 추이

기능별 요구기능 수에 대비하여 시험수를 파악하고, 각 시험에 대한 기능별 결함수에 따른 결함 우선순위별 해결 및 미해결 결함수의 정량적 데이터를 관리한다. 이에 따라 기능명, 요구기능수, 시험수, 결함발견수, 결함우선순위를 정의한다. <표 5-7>는 결함 우선순위별 발견 및 해결 결함수의 한 예를 나타낸다.

기능명	요구기능수	시험수		결함발견수	해결결함수	차기버전수	미해결결함수	Severity			
								Critical	Major	Average	Minor
A	100	128	6440	455	455	35	0	28	145	206	76
							해결수	28	145	206	76
							미해결수	0	0	0	0
B	21	104	3238	229	229	16	0	28	98	74	31
							해결수	28	98	74	31
							미해결수	0	0	0	0
C	55	450	9444	667	667	53	0	56	430	182	19
							해결수	56	430	182	19
							미해결수	0	0	0	0
기능 소계	176	682	19122	1351	1351	104	0	110	673	442	126
							해결수	110	673	442	126
							미해결수	0	0	0	0

<표 5-7> 결함 우선 순위별 발견 및 해결수

5.7.5 요구기능별 결함 해결율

요구 기능별로 결함에 따른 결함 해결율과 구현 여부를 정량적 데이터로 관리한다. 요구기능별 결함 발견수, 해결결함수, 미해결결함수, 각 요구기능별 결함 해결율, 그리고 구현 여부를 정의하도록 한다. <표 5-8>은 요구기능별 결함 해결율의 한 예를 나타낸다.

#	결함발견	해결결함	미해결결함	0%	25%	50%	75%	100%	구현여부
1	0	0	0	O					NT
2	4	2	2			O			
3	4	3	1				O		
4	0	0	0					O	O
5	0	0	0	O					NT
6	1	1	0					O	O
7	0	0	0	O					
8	4	1	3		O				
9	0	0	0	O					NT
10	13	0	13	O					
11	0	0	0	O					NT
12	0	0	0	O					NT
13	1	0	1	O					

<표 5-8> 요구기능별 결함 해결율

<용어설명>
#: 요구기능번호

5.7.6 요구 기능별 구현율

요구사항을 가장 구체적으로 표현하고 있는 기능에 대한 구현율을 표현하고자 하며, 이는 요구기능별 요구기능 수 대비 구현수, 요구기능 수 대비 미구현수, 이에 따른 구현율을 정량화하여 나타낸다. <표 5-9>는 요구기능별 구현율의 한 예를 나타낸다.

기능명	구분	요구기능수	구현수	미구현수	구현율
A	시험구현율	106	31	75	29.2
	개발구현율	91	31	60	34.1
	Defect해결율	91	31	60	48.6
B	시험구현율	40	12	28	30.00%
	개발구현율	39	12	27	30.77%
	Defect해결율	39	12	27	45.51%
C	시험구현율	111	19	92	17.1
	개발구현율	61	19	42	31.1
	Defect해결율	61	19	42	46.7
계	시험구현율	257	62	195	25.43%
	개발구현율	191	62	129	31.99%
	Defect해결율	191	62	129	46.94%

<표 5-9> 요구기능별 구현율

6. 결론

본 연구에서 제안한 품질 측정 프로세스 관리 방법은 Windows Application에 대하여 사전에 적용이 된 사례를 근간으로, Embedded SW에 적용 가능하도록 Embedded SW 및 결함 특성을 분석하여 관리 방법을 제안하였다.

본 연구에서 전개한 내용을 요약하여 살펴보면, 첫째로, Embedded 소프트웨어의 특성에 따른 결함 특성을 살펴보았으며, 두번째로, 소프트웨어의 결함 정의, 속성, 관리수준을 살펴보고, CMM에서 제시하는 결함 측정 요건과 데이터 활용 방법을 정의하였다. 세번째로, 결함 데이터를 이용한 정량적 관리 목표 수립, 관리 기준, 관리 방법을 알아보았다. 마지막으로, 결함 데이터 관리 절차 및 방법을 제안하였다.

이러한 본 연구의 내용은 크게 세 가지의 큰 의의가 있다. 첫째는, 기존 Application에 적용한 사례를 통해 Embedded SW에서도 적용할 수 있도록 Embedded SW 및 결함 특성을 분석하고 결함 데이터를 통해 정량적으로 프로세스를 관리하는 방법을 제안하였으며, 두번째로, 요구기능별, 일자별로 분석하여 테스트 단계에서 결함 데이터를 활용한 정량적 관리 방법을 제시하여, 전 단계(Whole lifecycle)에서 확대 적용 가능하도록 하였으며, 세번째로, 누적 결함 추이 분석을 통해 예측 가능한 잔존 결함을 파악하여 향후 예측 가능한 관리가 가능하도록 하였다.

하지만, 본 연구에서 보완하여야 할 점도 있다. 먼저, Embedded SW 결함 특성과 ISO9126과의 연관성을 규명하지 않았고, 두번째로, Embedded SW에 대하여는 실제 적용을 통한 효과성 검증이 없어, 향후 관련 프로젝트를 통해 효과성 검증이 필요하며, 세번째로, 사전에 예측된 결함 데이터를 근거로 하는 프로젝트의 경우에는 본 연구에서 제안한 내용 이외에 결함 예측 방법을 추가로 도입하여야 하는 문제점이 있다.

따라서, 향후에는 먼저, Embedded SW 결함 특성과 ISO9126과의 연관성을 체계적으로 연구하도록 할 예정이며, 두번째로, 결함수 예측과 결함제거 목표를 근간으로 하고, 결함수를 통한 프로세스 시작/종료 기준을 포함하는 품질 측정 프로세스 관리에 대하여 보다 발전된 연구 내용을 소개하고 싶다. 세번째로, 예상 결함 수를 정립하고, 프로세스의 종료 기준을 수립하여, 단계별 예상 결함 수 대비 실제 결함 수, 단계별 결함 밀도 추이, 일별 결함 발생 추이 등을 보다 발전적으로 연구할 예정이며, 네번째로, 품질 측정 프로세스 관리를 하는 방법의 효과성을 파악하기 위한 기준 및 실제 효과성을 제시하고자 한다.

이제, 본 연구를 마치면서 프로세스의 성숙도를 높이기 위한 중요한 목적 중에 하나인 결함 예방 및 제거 노력에 본 연구의 내용이 일조가 되길 기대하며, 최종 릴리즈 후에 발생하는 결함이 무결함(Defect Free)이 되는 이상적인 목표를 달성하기 위해 앞으로 소프트웨어 개발 단계 초기부터 결함 데이터를 적용하고 시작하였으면 하며, 따라서 이러한 본 연구를 토대로 향후 초기 단계에 결함 데이터를 이용한 품질 측정 프로세스 관리에 적용되길 기대한다.

참고 문헌

[1] Mark C. Paulk, Bill Curtis, Mary Beth Chrissis, Charles V. Weber, "Capability Maturity Model for Software", *SEI*, CMU/SEI-93-TR-024, February 1993.

[2] John H. Baumert, "Software Measures and the Capability Maturity Model", CMU/SEI-92-TR-25, Pittsburgh, Pa., *Software Engineering Institute, Carnegie Mellon University*, 1992.

[3] 김수동, "ISO9001 Compliant 객체지향 소프트웨어 품질시스템", *정보과학회지 제13권 제9호*, pp.27~45, 1995.9.

[4] David N. Card and Clark E. Humphrey, "Using Defect Data to Achieve SW-CMM Level 4", *2001 SEPG Conference, Software Engineering Institute*, March 2001

[5] Boehm, B. W., and others, "Characteristics of Software Quality," *TRW Software Series*, December 22, 1973.

[6] Murine, G. E., "Improving Management Visibility Through the Use of Software Quality Metrics," *Proceedings from IEEE Computer Society's Seventh*

International Computer Software & Application Conference. New York, N.Y., Institute of Electrical and Electronic Engineers, Inc., 1983.

[7] Jan Rabaey, "Digital Integrated Circuits-A Design Perspective", *Prentice Hall*, 1997

[8] Hermann Kopetz, "Real-Time Systems:Design Principles for Distrbiuted Embedded Applications", Kluwer Academic Publisher, 1997.

[9] Wayne Wolf, "Computers as 'Components'", *Morgan Kaufmann Publisher*, 2001.

[10] Alan Clements, "Microprocessor Systems Design-68000 Hardware, Software, and Interfacing(3rd Ed.)", *PWS Publisher*, 1998.

[11] 정기석 외 1 인, "내장형 시스템 설계:개론", *한국정보과학회*, July, 2002.

[12] Gaffney, J. and Robert Cruickshank, Richard Werling, Henry Fenry Felber, "Software Measurement Guidebook", *International Thomson Computer Press*, Boston, 1995.

[13] Kan, S. H., "Metrics and Models in Software Quality Engineering", *Addison Wesley*, 1995.

[14] William A. Florac, "Software Quality Measurement: A Framework for Counting Problems and Defects", *CMU/SEI-92-TR-22, Pittsburgh, Pa., Software Engineering Institute, Carnegie Mellon University*, 1992.

[15] Fenton, Norman E., "Software Metrics: A Rigorous Approach". *New York, N.Y., Van Nostrand Reinhold*, 1991.

[16] 조한상,안유환,박복남 외 1 인, "관리 척도를 이용한 CMM 핵심 프로세스 영역 측정 및 분석", *정보처리학회*, 2001.

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

[18] 김대영외 4 인, "*실시간운영체제연구개발동향*", 대한전자공학회, 2002.09.

[19] Robert E. Park, "Goal-Driven SW Measurement-Guidebook", *Software Engineering Institute*, Aug., 1996.