Journal of The Korea Society of Computer and Information
Vol. 17 No. 6 , June 2012

2012-17-6-3-6

결점 필터링 개념 기반 품질관리 노력 추정 모델

이 상 운*

An Quality Management Effort Estimation Model Based on Defect Filtering Concept

Sang-Un, Lee*

요 약

고 품질의 소프트웨어를 개발하기 위해서는 소프트웨어 내에 잠재된 결점을 제거하기 위한 품질관리계획이 요구된다. 이를 위해 결점 제거 프로필을 적절히 기술해야만 한다. 기존의 유조와 도관 모델은 결점이 제거되고 도피하는 결점을 계산하는데 복잡한 과정을 수행한다. 또한 어느 단계에서 삽입된 결점이 제거되고 도피하였으며, 단계별 결점 발견율이 얼마인지를 상세히 알고 있어야만 한다. 이러한 복잡한 과정을 단순화하기 위해 본 논문은 결점 필터링 개념에 기반하여 모델을 제시하였다. 제시된 모델은 임의의 단계에서 제거와 도피한 결점이 어느 단계에서 삽입된 결점에 관련되어 있는지 고려할 필요가 없어 결점을 보다 간략히 기술할 수 있는 장점이 있다. 또한, 결점 제거 품질척도와 생산성 척도의 함수에 기반하여 결점제거에 요구되는 노력 추정 모델을 제시하였다.

▸ 키워드 : 결점 필터링, 유조와 도관 모델, 품질계획, 결점제거 효과성, 결점제거 효율성

Abstract

To develop high quality software, quality control plan is required about fault correction that is latent within software. We should describe fault correction profile properly for this. The tank and pipe model performs complex processes to calculate fault that is remove and escapes. Also, we have to know in which phase the faults were inserted, removed and escaped and know the fault detection rate at any phases. To simplify such complex process, this paper presented model to fault filtering concept. Presented model has advantage that can describe fault more shortly because need not to consider whether was involved in fault that escaped fault is inserted at any step at free

• 제1저자 : 이상운
• 투고일 : 2012. 02. 21, 심사일 : 2012. 03. 20, 게재확정일 : 2012. 05. 03.
* 강릉원주대학교 멀티미디어공학과 (Dept. of Multimedia Eng., Gangneung-Wonju National University)

step. Also, presented effort estimating model that do fetters in function of fault removal quality and productivity measure and is required in fault detection.

▸ Keywords : Defect Filtering, Tank and Pipe Model, Quality Planning, Defect Removal Effectiveness, Defect Removal Efficiency

I. 서 론

일반적으로 소프트웨어에 남아 있는 모든 결점을 검출하여 제거할 때까지 시험을 무한정 수행할 수는 없다. 대부분은 할당된 자원(시간, 예산 또는 시험 케이스)이 고갈되었을 때, 또는 사전에 명시된 신뢰수준을 만족시키거나 계속 시험을 진행함으로써 얻는 이득이 없을 경우 시험을 종료하는 규칙을 적용한다[1]. 따라서, 시험 종료시점을 예측하기 위해서는 결점 수, 결점 검출율, 결점 제거 노력, 결점 제거 시간 데이터가 필요하며 이들 데이터를 이용해 소프트웨어 품질관리 계획을 수립하여 실행에 옮길 수 있다.

소프트웨어에 삽입된 결점 수를 예측하기 위해서는 제품의 규모와 결점 삽입율이 필요하며, 결점제거 노력과 시간은 시험을 종료하는데 필요한 노력을 평가하는데 사용된다.[2] 결점 삽입과 제거 현상에 대한 모형으로는 유조와 도관 모형에 기반한 연쇄수율 개념 기반 모델과 결점 필터링 개념 기반 모델이 있다.[3,4] 연쇄수율 기반 모형은 임의의 단계에서 결점 검출시 해당 결점이 어느 단계에서 삽입된 결점인지를 파악하여 결점 삽입 단계별 수율을 계산하여야만 한다. 따라서 이 모형을 적용시 제거되는 결점 수나 발견이 되지 않고 다음 단계로 도피하는 결점 수를 산출하는데 매우 복잡한 과정을 거쳐야 한다. 이에 비해 결점 필터링 기반 모형은 제거되는 결점이 어느 단계에서 삽입되었는가를 분석할 필요가 없으며, 임의의 단계에서 볼 때 단지 이전 단계에서 도피한 결점과 이 단계에서 제거된 결점만을 고려함으로써 매우 단순하여 쉽게 적용이 가능하다. 그러나 결점 필터링 기반 모형도 결점이 삽입되고 제거되는 단계를 명확히 고려하지 않았으며, 제거되는 결점 수와 검출되지 않고 다음 단계로 도피하는 결점 수를 계산하는 수식을 보다 단순화시키지 못한 단점을 갖고 있다.

본 논문은 먼저, 결점 필터링 기반 모형에 대해 결점 삽입과 제거 단계를 명확히 설정하고, 제거되는 결점과 도피하는 결점을 보다 쉽게 계산할 수 있는 모형을 제시한다. 이어서, 품질척도와 생산성 척도가 상수가 아닌 어떠한 함수 형태를 취한다는 가정하에 이 함수에 기반하여 결점 제거에 소요되는 노력을 추정할 수 있는 모델을 제안한다.

Ⅱ장에서는 관련 연구와 문제점을 살펴보고, Ⅲ장에서는 품질계획 모델을 제시한다. Ⅳ장에서는 제안된 모델을 이용해 품질척도와 남아 있는 결점 수를 산출할 수 있음을 보이고 이어서 품질계획에 소요되는 노력 추정 모델의 성능을 평가한다.

II. 결점 프로파일 관련연구와 문제점

소프트웨어 품질을 계획하고 관리하기 위해서는 결점이 소프트웨어에 삽입되고 제거되는 과정을 이해하여야만 한다. 결점 삽입과 제거 과정은 Jones[5]이 제안한 그림 1의 유조와 도관 모형(Tank and Pipe Model)에 기반하여 Boehm[6]이 제안한 COQUALMO (COnstructive QUALity MOdel)로 설명될 수 있다.[7]

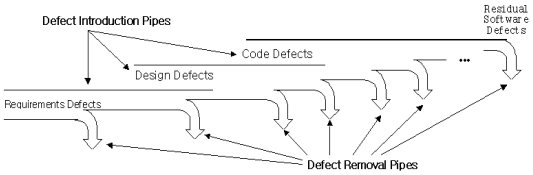


그림 1. 유조와 도관 모형 기반 결점 모델
Fig. 1. Defect Model based on Tank and Pipe Model

요구사항 분석, 설계와 코딩 단계를 수행하면서 결점들이 유조에 유입된다. 유입된 결점들은 해당 단계부터 시작하여 이후에 수행되는 단계들에서의 검토와 시험 과정에서 도관을 통해 제거된다. 즉, 요구사항 결점은 요구사항 분석 단계에서 유조에 유입되어 요구사항 분석 검토부터 시작하여 개발이 종료될 때까지 수행되는 모든 단계들에서의 검토와 시험 과정에서 발견되어 도관을 통해 제거된다. 설계 결점과 코딩 결점도 각각 설계단계와 코딩단계에서 유조에 유입된 후 요구사항 결점과 동일한 방법으로 이후 단계에서의 검토와 시험과정에서 도관을 통해 제거된다.

1. 결점 삽입 모델

Chulani[8]은 COCOMO 모델의 노력(Effort, E)을 추

정하는 $E=a\times Size^b$나 $E=a\times Size^b\times\prod_{i=1}^{15}$노력승수 (Effort Multiplier)에 기반하여 삽입 결점 (Defect Injected, DI)에 대해 식 (1)의 모델을 제안하였다. 노력 승수(또는 비용 인자)는 COCOMO II[9]에 정의되어 있다.

$$DI_j = DIR_{Baseline\,j}\times Size^b \quad (1)$$

여기서 j는 요구사항 명세서, 요구사항분석단계, 설계단계와 코딩단계이다. $DIR_{Baseline\,j}$은 결점 삽입율 (Defect Injected Rate)로 요구사항분석 단계는 5/KDSI (Thousands of Delivered Source Instructions), 설계단계는 25/KDSI, 코딩단계는 15/KDSI, 요구사항명세서는 15/KDSI이다. 이 식으로부터 삽입된 결점에 대한 추정 값은 식 (2)로 계산된다. 여기서, 여기서 j는 요구사항분석 단계, 설계단계와 코딩단계로 분류되며, a_j는 상수, QAF (Quality Adjustment Factor)는 $\prod_{i=1}^{22} DIR-driver$이다.

$$DI_{Est\,j} = a_j\times DI_j\times QAF_j$$
$$= a_j\times DIR_{\square\,|\,j}\times Size^b\times QAF_j \quad (2)$$

Chulani[7]는 COQUALMO의 소프트웨어 결점 밀도 예측 모델에서 식 (3)을 제시하였다. 식 (3)은 식 (2)와 유사함을 알 수 있다.

$$DI_{Est,j} = a_j\times Size^{b_j}\times QAF_j \quad (3)$$

2. 결점 제거 척도

결점 제거 척도(Metric)에 대한 연구는 표 1에 제시하였다.

표 1. 결점 제거 척도
Table 1. Defect Removal Metrics

구분	품질 척도	정의
Jalote와 Saha[4]	Defect removal efficiency(DRE)	$\frac{\text{Number of defects detected at phase i}}{\text{Number of defects actually present in phase i}}$
Walton [10]	Verification Effectiveness(VE)	$\frac{\text{Defects found by the process}}{\text{Defects presented to the process}}$ $\text{Presented(i)} = \text{Injected (i)} + \text{escaped (i-1)}$
Westfall [11]	Defect removal effectiveness (or Efficiency)	$\frac{\text{Defects removed during a development phase}}{\text{Defect latant in the product at that phase}}$
Weller [2,12]	Effectiveness	$\frac{\text{Defect removed(i)}}{\text{Defect Injected(i)} + \text{Defect Escaped(i-1)}}$
Roy[3]	Process Yield	$\frac{\text{Defects removes in the stage}}{\text{Defects present at stage entry}} * 100\%$
Jose, Anju와 Pillai[13]	Defect Removal Efficiency(DRE)	$\frac{\text{removed at phase i}}{\text{escaped from phase i-1} + \text{injected during phase i}}$
UKSMA [14]	Defect Removal Efficiency(DRE)	$\frac{\text{Number of defects removed}}{\text{Number of defects inserted during a particular stage}}$
Koch[15]	Defect removal yield(DRY)	$\frac{\text{Number of defects removed in phase i}}{\text{Number of defects existing at phase i}} \times 100\%$

구분	생산성 척도	정의
Weller [2,12]	Efficiency	$\frac{\text{Defect removed(i)}}{\text{Effort expended(i)}}$
Koch[15]	Defect removal efficiency(DRE)	$\frac{\text{Number of defects removed in phase i}}{\text{Effort expended in phase i}}$

Weller[2,12]와 Thakur[16]는 Effectiveness를 품질척도 (Quality Metric)로 Efficiency를 생산성 척도 (Productivity Metric)라 정의하였다. 표 1의 품질척도 부분을 살펴보면 Effectiveness (VE 1회, DRE 2회), Yield (DRY 1회, Process Yield 1회)와 Efficiency (DRE 4회) 등 상이한 용어로 거론하고 있다. 이들 혼용된 용어를 Weller[2,12]와 Thakur[16]의 정의에 기반하여 DRE (Defect Removal Effectiveness)로 통일시키고 이를 DRE_q라 하자. 품질 척도의 정의 부분을 살펴보면 detected(1회), founded (1회)와 removed(6회)를 모두 removed로 통일시켜 해석할 수 있다. 또한, existing(1회), presented(2회), latent(1회)와 inserted(1회)를 escaped + injected(2회) 개념으로 해석하면 품질척도는 식 (4)로 정의될 수 있다.

$$\text{품질척도 } (DRE_q) = Defect\ Removal\ Effectiveness$$
$$= \frac{Defect\ removed(i)}{Defect\ Injected(i) + Defect\ Escaped(i-1)} \quad (4)$$

생산성 척도는 Weller[2,12]와 Thakur[16]가 정의한 바와 같이 DRE (Defect Removal Efficiency)로 정의할 수 있으며, 이를 DRE_p라 하면 식 (5)로 표현된다.

$$\text{생산성척도}(DRE_p) = Defect\ Removal\ Efficiency$$
$$= \frac{Defect\ removed(i)}{Effort\ expended(i)} \quad (5)$$

3. 결점 제거 모델

3.1 연쇄수율 개념 기반 모델

Roy[3]는 유조와 도관 모형에 기반하여 결점 삽입 도관을 통해 요구사항 분석, 설계와 코딩 단계에서 발생하는 결점들이 유조에 유입되어 후속 단계에서 수행되는 검토와 시험을 거치면서 결점 제거 도관을 통해 제거되는 과정은 그림 2의 연쇄 수율 (Chain Yield)로 해석될 수 있다고 제시하였다. 여기서, 수율은 Roy[3]의 Process Yield 또는 Koch[15]의 DRY로 계산된다.

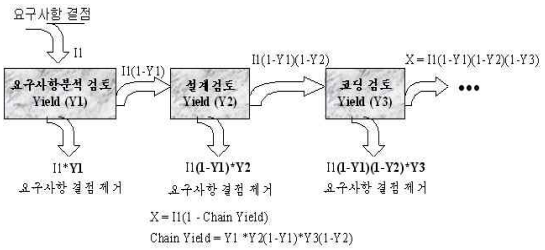
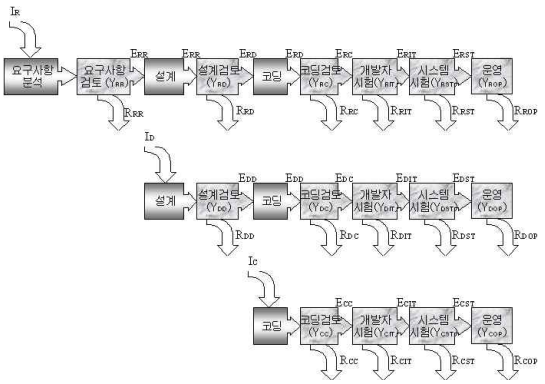


그림 2. 요구사항 결점의 연쇄적 제거 과정
Fig. 2. Chained Removal Process for Requirements Defects

그림 2의 연쇄 수율 기반 제거되는 결점 수는 그림 3으로 해석할 수 있다. 여기서 I는 삽입된 (Injected) 결점 수, R은 제거된 (Removed) 결점 수, E는 도피한 (Escaped) 결점 수를 의미하며, Y는 수율 (Yield)이다.

여기서는 소프트웨어 생명주기의 단계를 전형적인 폭포수 모델에 기반하고 있다. 예로, 요구사항분석 (R), 설계 (D), 코딩 (C), 단위시험 (UT), 통합시험 (IT), 시스템시험 (ST), 수락시험 (AT), 운영 (OP)으로 분류한다.



[요구사항 결점]

$$R_{RR} = I_R \cdot Y_{RR}$$
$$E_{RR} = I_R(1 - Y_{RR})$$
$$R_{RD} = E_{RR} \cdot Y_{RD} = I_R(1 - Y_{RR}) \cdot Y_{RD}$$
$$E_{RD} = E_{RR}(1 - Y_{RD}) = I_R(1 - Y_{RR})(1 - Y_{RD})$$
$$R_{RC} = E_{RD} \cdot Y_{RC} = I_R(1 - Y_{RR})(1 - Y_{RD}) \cdot Y_{RC}$$
$$E_{RC} = E_{RD}(1 - Y_{RC}) = I_R(1 - Y_{RR})(1 - Y_{RD})(1 - Y_{RC})$$
$$R_{RIT} = E_{RC} \cdot Y_{RIT} = I_R(1 - Y_{RR})(1 - Y_{RD})(1 - Y_{RC}) \cdot Y_{RIT}$$
$$E_{RIT} = E_{RC}(1 - Y_{RIT}) = I_R(1 - Y_{RR})(1 - Y_{RD})(1 - Y_{RC})(1 - Y_{RIT})$$
$$R_{RST} = E_{RIT} \cdot Y_{RST} = I_R(1 - Y_{RR})(1 - Y_{RD})(1 - Y_{RC})(1 - Y_{RIT}) \cdot Y_{RST}$$
$$E_{RST} = E_{RIT}(1 - Y_{RST}) = I_R(1 - Y_{RR})(1 - Y_{RD})(1 - Y_{RC})(1 - Y_{RIT})(1 - Y_{RST})$$
$$R_{ROP} = E_{RST} \cdot Y_{ROP} = I_R(1 - Y_{RR})(1 - Y_{RD})(1 - Y_{RC})(1 - Y_{RIT})(1 - Y_{RST}) \cdot Y_{ROP}$$

[설계 결점]

$$R_{DD} = I_D \cdot Y_{DD}$$
$$E_{DD} = I_D(1 - Y_{DD})$$
$$R_{DC} = E_{DD} \cdot Y_{DC} = I_D(1 - Y_{DD}) \cdot Y_{DC}$$
$$E_{DC} = E_{DD}(1 - Y_{DC}) = I_D(1 - Y_{DD})(1 - Y_{DC})$$
$$R_{DIT} = E_{DC} \cdot Y_{DIT} = I_D(1 - Y_{DD})(1 - Y_{DC}) \cdot Y_{DIT}$$
$$E_{DIT} = E_{DC}(1 - Y_{DIT}) = I_D(1 - Y_{DD})(1 - Y_{DC})(1 - Y_{DIT})$$
$$R_{DST} = E_{DIT} \cdot Y_{DST} = I_D(1 - Y_{DD})(1 - Y_{DC})(1 - Y_{DIT}) \cdot Y_{DST}$$
$$E_{DST} = E_{DIT}(1 - Y_{DST}) = I_D(1 - Y_{DD})(1 - Y_{DC})(1 - Y_{DIT})(1 - Y_{DST})$$
$$R_{DOP} = E_{DST} \cdot Y_{DOP} = I_D(1 - Y_{DD})(1 - Y_{DC})(1 - Y_{DIT})(1 - Y_{DST}) \cdot Y_{DOP}$$

[코딩 결점]

$$I_C = 300$$
$$R_{CC} = I_C \cdot Y_{CC}$$
$$E_{CC} = I_C(1 - Y_{CC})$$
$$R_{CIT} = E_{CC} \cdot Y_{CIT} = I_C(1 - Y_{CC}) \cdot Y_{CIT}$$
$$E_{CIT} = E_{CC}(1 - Y_{CIT}) = I_C(1 - Y_{CC})(1 - Y_{CIT})$$
$$R_{CST} = E_{CIT} \cdot Y_{CST} = I_C(1 - Y_{CC})(1 - Y_{CIT}) \cdot Y_{CST}$$
$$E_{CST} = E_{CIT}(1 - Y_{CST}) = I_C(1 - Y_{CC})(1 - Y_{CIT})(1 - Y_{CST})$$
$$R_{COP} = E_{CST} \cdot Y_{COP} = I_C(1 - Y_{CC})(1 - Y_{CIT})(1 - Y_{CST}) \cdot Y_{COP}$$

그림 3. 연쇄 수율에 기반한 결점 삽입과 제거 모델 해석
Fig. 3. Analysis of Defect Insert and Removal Model based on Chain Yield

Roy[3]의 연쇄수율 기반 모델을 적용할 경우, 요구사항, 설계와 코딩단계에서 삽입된 결점 각각이 후속 단계에서 제거되는 과정을 통해 얻어지는 수율을 계산하고 그림 3에 기반하여 제거되는 결점과 도피하는 결점을 계산해야 최종적으로 남아 있는 결점 수를 계산할 수 있다. 즉, 이 모델을 적용하기 위해서는 하나의 단계에서 삽입된 결점이 후속 단계에서 얼마나 제거되는지, 수율은 얼마인지를 모두 알아야 하는 복잡성을 갖고 있다. 실제로 제거된 결점이 어느 단계에서 삽입되어 단계가 진행될수록 얼마만큼씩 제거시킬 수 있느냐가 중요한 것이 아니다. 이 보다는 어느 단계에서 삽입된 결점이 제거되었는지가 아니라 임의의 단계에서 볼 때, 지금까지 남아 있는 결점수 중에서 얼마나 많이 제거하였는가가 보다 중요하게 활용될 수 있다.

3.2 결점 필터링 개념 기반 모델

Jalote와 Saha[4]는 그림 4와 같이 요구사항분석, 설계와 코딩단계에서 결점이 삽입되고 각 단계의 검토와 시험단계에서 결점이 제거된다는 가정하에 제거 결점을 간략하게 계산하였다. 여기서 DRE_q는 Roy[3]가 제안한 수율 Y_i에 해당되며, 삽입 결점 수는 IJ_i로 표기되어 있다.

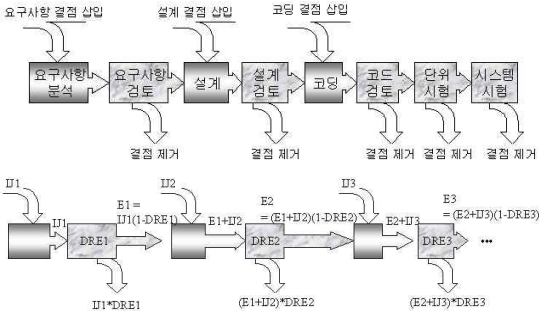


그림 4. 결점 삽입과 제거
Fig. 4. Defect Insert and Removal

이 모델은 결점 필터링 (Defect Filtering) 개념으로 설명될 수 있다. 결점 필터링이란 소프트웨어를 개발하는 과정 중에서 임의의 한 단계를 기준으로 할 때 그림 5와 같이 이전 단계에서 제거되지 않고 도피한 결점이 현재 수행되는 단계에 상속되며, 이 단계를 진행하는 과정에서 삽입된 결점과 합하여 현 단계에서의 잔존하는 (Latent, L) 결점이 된다. 잔존 결점 수는 검토와 시험 과정을 거치면서 일부분이 제거되고 남아 있는 결점은 다음 단계로 도피하는 것으로 모형화된다.

즉, i번째 단계에서 잔존하는 결점 수 L_i는 $i-1$번째 단계에서 제거되지 않고 도피한 결점 수 $E_{i-1}+i$ 번째 단계에서 삽입된 결점 수 I_i가 된다. L_i의 결점 수에 대해 검토/시험 과정의 품질척도인 $DRE_{q(i)}$를 곱하면 제거된 결점 수 R_i가 산출되고, 제거되지 않은 결점 수 E_i는 $i+1$ 단계로 도피한다.

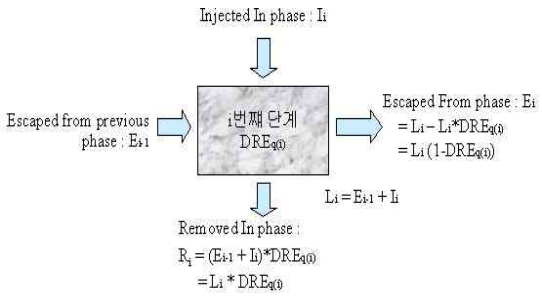


그림 5. 결점 필터링
Fig. 5. Defect Filtering

지금까지 살펴본 바와 같이 결점 삽입과 제거 현상의 모형화에 대한 연구는 Roy[3]의 그림 3과 Jalote와 Saha[4]의 그림 4에서 알 수 있듯이 어떤 단계에서 결점이 제거되는지에 대한 단계가 명확히 구분되어 있지 않고 일반화된 모델도 제시되지 않고 있다. 따라서 Ⅲ장에서는 연쇄 수율 개념 기반이 아닌 결점 필터링 개념에 기반한 일반화된 결점 제거 모델과 더불어 결점 제거에 소요되는 노력(또는 비용)을 계획하기 위한 품질계획 모델을 제시한다.

Ⅲ. 결점 필터링 개념 기반 품질계획 모델

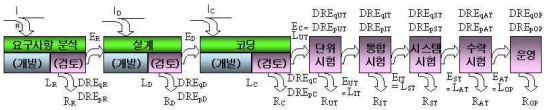
1. 결점 필터링 개념 기반 일반화된 품질계획 모델

Jalote와 Saha[4]가 제안한 모델에 기반하면 소프트웨어 개발 단계와 결점 삽입과 제거 현상은 다음과 같이 가정할 수 있다.

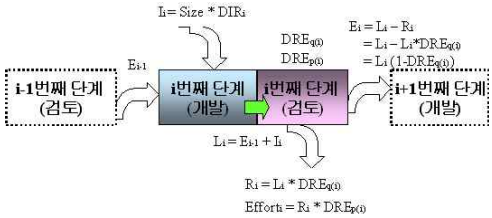
[가정 1]요구사항 분석, 설계와 코딩단계에서는 검토가 수행되고 시험은 단위, 통합, 시스템과 수락시험단계로 구분되며, 운영단계 까지를 고려하여 결점 삽입과 제거에 대한 생명주기를 결정한다.

[가정 2]검토, 시험과정에서는 결점이 삽입되지 않는다.

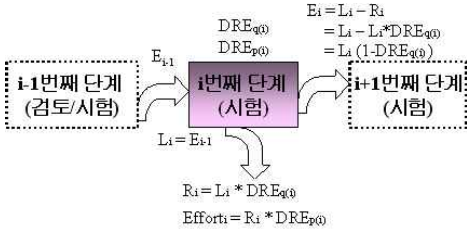
위 가정에 의해 요구사항 분석, 설계와 코딩 단계에서는 결점 삽입과 제거가 발생하며, 단위시험부터 운영단계까지는 결점 제거 현상만이 발생한다. Jalote와 Saha[4]이 제안한 그림 4 모델에서 단계를 i, IJ_i를 I_i, $L_i = E_{i-1} + I_i$, DRE_i를 DRE_{qi}로 치환하면 임의의 단계에서의 잔존, 제거와 도피 결점 수는 그림 6에 의거하여 표 2와 같이 표현된다.



(a) 개발 단계별 결점 삽입과 제거 모형



(b) 개발과 검토가 수행되는 단계



(c) 시험과 운영단계

그림 6. 단계별 결점 삽입과 제거 모형
Fig. 6. Defect Insert and Removal Model based on Phases

표 2. 결점 필터링 개념 적용시 일반화된 제거와 도피 결점 수
Table 2. Generalized Removal and Escaped Number of Defects based on Defect Filtering Concept

단계	결점 구분		
	잔존 결점 (L_i)	제거 결점 (R_i)	도피 결점 (E_i)
요구사항 검토 (R)	I_R	$L_R \cdot DRE_{qR}$	$L_R(1-DRE_{qR})$
설계 검토 (D)	E_R+I_D	$L_D \cdot DRE_{qD}$	$L_D(1-DRE_{qD})$
코딩 검토 (C)	E_D+I_C	$L_C \cdot DRE_{qC}$	$L_C(1-DRE_{qC})$
단위 시험 (UT)	E_C	$L_{UT} \cdot DRE_{qUT}$	$L_{UT}(1-DRE_{qUT})$
통합 시험 (IT)	E_{UT}	$L_{IT} \cdot DRE_{qIT}$	$L_{IT}(1-DRE_{qIT})$
시스템 시험 (ST)	E_{IT}	$L_{ST} \cdot DRE_{qST}$	$L_{ST}(1-DRE_{qST})$
수락시험 (AT)	E_{ST}	$L_{AT} \cdot DRE_{qAT}$	$L_{AT}(1-DRE_{qAT})$
운영 (OP)	E_{AT}	$L_{OP} \cdot DRE_{qOP}$	$L_{OP}(1-DRE_{qOP})$

표 2에 기반하면 식 (6)의 모델을 유도할 수 있다.

$$I_i = Size * DIR_i, \quad DRE_{q(i)} = \frac{R_i}{L_i}$$

$$L_i = E_{i-1} + I_i \begin{pmatrix} i=1, E_{i-1}=0 \\ i \geq 4, I_i = 0 \end{pmatrix}, i = 1.2.\cdots, 8$$

$$E_i = L_i(1-DRE_{q(i)}), \quad R_i = L_i * DRE_{q(i)}$$

$$DRE_{p(i)} = \frac{R_i}{Effort_i}, \quad Effort_i = R_i * DRE_{p(i)} \qquad (6)$$

2. 품질과 생산성 척도 기반 품질관리 노력 추정 모델

소프트웨어 결점을 계획하고 관리하기 위해서는 각 단계에서의 결점 삽입율, 결점 제거율과 더불어 검토나 시험에 소요되는 노력(또는 비용)의 양을 결정해야 한다. 그러나 DRE_q와 DRE_p가 단계별로 어떠한 경향을 보이고 있는지에 대한 연구는 수행되지 않고 있다. 따라서 추가로 다음 가정을 제시한다.

[가정 3]일반적으로 검토나 시험이 진행될수록 소프트웨어 내에 잔존하고 있는 결점들이 검출되어 제거됨으로서 결함 제거율(DRE_q)은 감소하며, 남아 있는 결점들의 감소로 인해 하나의 결점을 검출하는데 필요한 시간(DRE_p)는 점차적으로 증가하는 경향을 보인다.

[가정 4] DRE_q와 DRE_p는 함수 형태로 표현되며, 단계마다 다른 값을 가진다. 단, 임의의 단계에서는 소프트웨어의 규모와 복잡도에 관계없이 동일한 값을 갖는다.

[가정 4]에 기반하여 삽입 결점율 함수를 $f(DIR)$, 제거된 결점 수의 품질척도 함수를 $f(DRE_q)$, 생산성척도 함수를 $f(DRE_p)$라 하면 소프트웨어 개발 단계별로 결점 제거를 위해 검토/시험에 투입되는 노력(또는 비용)은 식 (7)을 얻을 수 있다.

$$\text{Effort}_i = L_i \cdot f(DRE_{q(i)}) \cdot f(DRE_{p(i)}) = R_i \cdot f(DRE_{p(i)}) \qquad (7)$$

$DRE_{q(i)}$, $DRE_{p(i)}$와 관련하여 수집된 노력 데이터는 표 3에 제시하였다. 여기서 [가정 3]에 일치하는 데이터로 Walton[10]의 데이터를 선택할 수 있다.

표 3. $DRE_{q(i)}$와 $DRE_{p(i)}$ 관련 품질계획 수행 노력 데이터
Table 3. Quality Planning Effort Data for $DRE_{q(i)}$ and $DRE_{p(i)}$

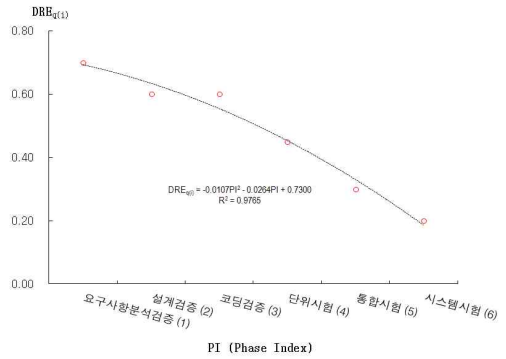
$DRE_{q(i)}$	요구사항 검토	설계 검토	코딩 검토	단위 시험	통합 시험	시스템 시험	수락 시험
Jalote et al.[4]	7/Page	10/Page	35/Kloc	8/Kloc	-	4/Kloc	-
Card[17]	0.00	0.45	0.50	0.50	-	0.75	-
Criscione et al.[18]	0.00	0.04	0.11	0.30	0.40	-	-
Walton[10]	0.70	0.60	0.60	0.45	0.30	0.20	-
Westfall[11]	0.63	0.60	0.55	0.60	-	-	-
Roy[3]	-	0.33	0.525	0.50	-	-	-
Jose et al.[13]	0.67	0.51	0.73	0.88	-	-	-
Longstreet[19]	0.15	0.30	0.20	0.25	-	-	-

$DRE_{p(i)}$	요구사항 검토	설계 검토	코딩 검토	단위 시험	통합 시험	시스템 시험	수락 시험
Jalote et al.[4] (Hrs)	2.0	7	24	87.1	-	333.4	-
Walton[10] (Hrs/Defect)	1.5	3.1	4.8	7.28	10.64	17.50	-

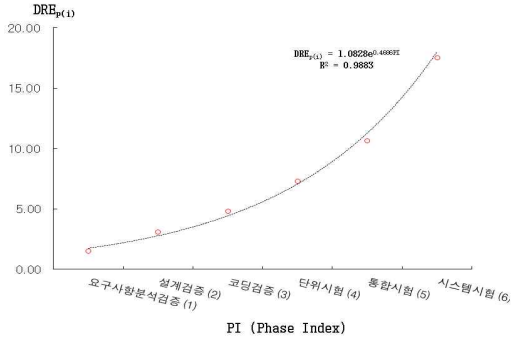
Walton[10]의 데이터를 이용하여 단계 인덱스에 따른 $DRE_{q(i)}$와 $DRE_{p(i)}$를 표현하면 그림 7과 같다. 각각의 데이터에 대해 회귀분석을 수행하여 얻은 모델은 식 (8)과 (9)에 제시되어 있다.

$$f(DRE_{q(i)}) = -1.0741\Pi^2 - 2.5429\Pi + 73.0000 \qquad (8)$$

$$f(DRE_{p(i)}) = 1.0828e^{0.4686\Pi} \qquad (9)$$



(a) 품질 척도 함수



(b) 생산성 척도 함수
그림 7. 결점 제거와 관련된 품질과 생산성 척도 함수
Fig. 7. Quality and Productivity Metric Function for Defect Removal

IV. 모델 평가

1. 품질척도와 노력 추정 가능

일반화된 결점 제거 모델 적용시 I_i와 R_i만 알고 있다면 $DRE_{q(i)}$, E_i와 L_i 계산이 가능하다. 또한, $DRE_{p(i)}$를 알고 있는 경우 $Effort_i$도 계산이 가능하다. 표 3의 논문들로부터 각 단계에서의 삽입과 제거되는 총 결점 수 만 추출하면 표 4를 얻는다.

표 4에 의거 그림 8과 같이 $DRE_{q(i)}$를 계산할 수 있다. 또한 그림 9의 투입 노력도 계산이 가능하다.

2. 품질노력 추정 모델 성능

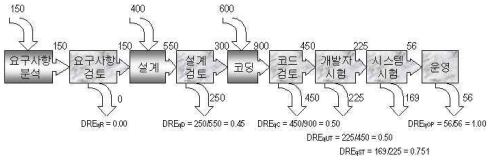
모델의 성능은 결정계수(Coefficient of determination, R^2)

표 4. 각 단계에서 삽입과 제거된 총 결점 수와 투입 노력
Table 4. The Total Number of Removal Defects and Effort for Each Phase

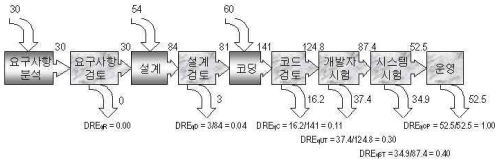
참고문헌	삽입 결점 수			제거 결점 수							
	요구사항분석	설계	코딩	요구사항 분석검토	설계 검토	코딩 검토	단위 시험	통합 시험	시스템 시험	수락 시험	운영
[17]	150	400	600	0	250	450	225	-	169	-	56
[18]	30	54	60	0	3	16.2	37.4	-	34.9	-	52.5
[10]	590	977	2548	413	687	1804	641	198	93	-	-
[11]	16	29	41	10	21	31	15			-	10
[13]	34	67	142	23	40	130	44			-	6

참고문헌	노력/규모							
	요구사항 분석검토	설계 검토	코딩 검토	단위 시험	통합 시험	시스템 시험	수락 시험	운영
[10]	1.50	3.10	4.80	7.28	10.64	17.50	-	-

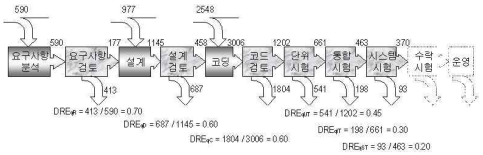
참고문헌 [17]



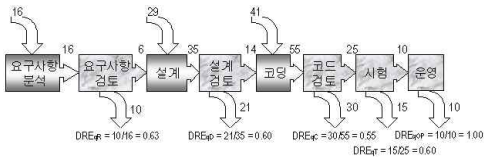
참고문헌 [18]



참고문헌 [10]



참고문헌 [11]



참고문헌 [13]

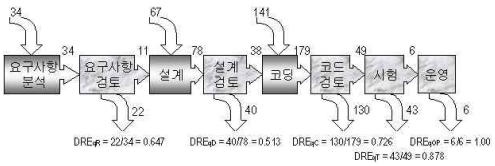


그림 8. I_i과 R_i로부터 $DRE_{q(i)}$ 계산
Fig. 8. Computation of $DRE_{q(i)}$ from I_i and R_i

로 평가한다. 종속변수의 값은 독립변수에 의해 결정되는 부분과 미지의 오차의 합으로 나타나며, 회귀직선에 의해 설명되는 변동 비율을 결정계수라 한다. 결정계수의 값이 클수록 쓸모 있는 회귀직선이 되지만 좋은 모델로 선정하기 위한 기준은 없는 실정이다.

참고문헌 [10]

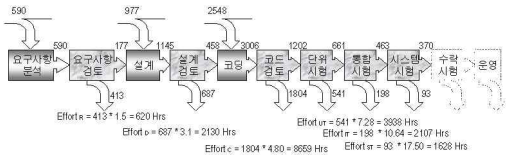


그림 9. $DRE_{p(i)}$로부터 $Effort_i$ 계산
Fig. 9. Computation of $Effort_i$ from $DRE_{p(i)}$

모델의 정확도를 판단하기 위해, Briand et al.[20]이 적용한 MMRE(Mean Magnitude of Relative Error)와 Conte et al.[21]와 Fenton[22]의 Pred(0.25)를 적용하였다. 모델의 정확도를 평가하는 MMRE는 작은 값이면 평균적으로 좋은 모델임을 알 수 있다. Pred(0.25)는 예측된 값의 MRE가 실측값의±25% 범위에 있는 개수를 의미한다. Conte et al.[21]과 Fenton[22]은 $MMRE \leq 0.25$(25% 이하)이면 개발비용을 예측하는 모델로 채택 가능하며, Pred(0.25)가 75% 이상일 때 좋은 예측모델로 될 수 있음을 제안하였다.

결점제거에 투입된 노력을 추정하는 식 (7)의 모델로 실측 데이터를 추정한 결과 얻은 모델의 성능은 표 5에 제시하였다. 또한, Pred(0.25)에 대해 실측 데이터와의 편차를 그림 10에 제시하였다.

제안된 결점제거 노력 추정 모델은 Conte et al.[21]이 제안한 $MMRE \leq 0.25$와 Pred(0.25)$\geq$75%의 기준 모두를 만족한다. 따라서, 새로운 소프트웨어 개발시 제안된 모델을 적용하면 결점제거 노력에 대한 품질관리 계획을 보다 현실성 있게 작성할 수 있을 것이다.

표 5. 결점제거노력 추정 모델 성능
Table 5. Performance of Defect Removal Effort Estimated Model

모델 성능	결정계수	MMRE(%)	Pred(0.25)
	0.9899	13.7633	1.0000 (100%)

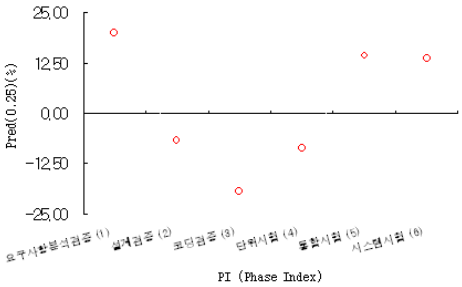


그림 10. 실측 데이터 예측 편차
Fig. 10. The Prediction Error for Actual Data

V. 결론 및 향후 연구과제

소프트웨어의 결점을 개발단계에서 적절히 제거하지 못하면 납품 후 사용자를 만족시키지 못해 프로젝트 실패 또는 고객의 외면을 받을 수 있다. 소프트웨어에 잠재된 결점을 개발 초기단계부터 되도록 많이 제거함으로써 품질관리에 소요되는 노력을 최적으로 관리할 수 있을 것이다.

기존의 소프트웨어 결점관리는 유조와 도관 모델에 기반하여 임의의 단계에서 발견되거나 발견되지 않고 다음 단계로 도피하는 결점을 계산하기 위해서는 어느 단계에서 삽입된 결점인지를 알아야 하며, 이 결점에 대해 해당 단계의 결점 제거 효율성에 기반하여 도피하는 결점을 계산하였다. 따라서, 매우 복잡한 과정을 거쳐야만 한다.

본 제안 모델은 어느 단계에서 삽입된 결점인지에 관계없이 임의의 단계에서 제거되거나 도피하는 결점 수 만을 고려함으로써 매우 간단한 방법으로 결점을 관리하는 방법을 제안하였다. 또한, 이 방법에 기반하여 임의의 단계에서 발견된 결점의 효율성인 품질척도와 결점 제거에 소요된 노력인 생산성 척도 함수에 기반하여 결점관리에 소요되는 노력을 도출할 수 있는 모델을 제시하였다. 따라서 제안된 모델은 품질관리에 소요되는 노력을 보다 효율적으로 계획하고 관리할 수 있도록 하였다.

그러나 본 논문에서는 품질관리에 소요되는 노력을 추정하는데 적용된 데이터를 충분히 확보하지 못해 일반적인 모델을 제시하지는 못하였다. 따라서 추후 다양한 품질척도와 생산성 척도 이력 데이터를 수집하여 일반화된 품질관리 모델 제시에 대한 연구를 수행할 예정이다.

참고문헌

[1] M. C. K. Yang, and A. Chao, "Comparisons of methods in reliability estimation and stopping rules for software testing", IEEE Trans. on Reliability, Vol. 44, pp. 315-321, 1995.

[2] E. Weller, "Defect Management in Development and Test", International Conference on Software Testing, Analysis & Review, San Jose, CA., 1999.

[3] D. M. Roy, "Synergy of Review Techniques from PSP(SM) to Formal Inspections", SEPG Conference, Phoeniz, AZ, 2002.

[4] P. Jalote and A. Saha, "Optimization of Quality Control Tasks in a Software Development Process", Department of Computer Science and Engineering, Indian Institute of Technology, Kanpur, India, 2002.

[5] C. Jones, "Programming Defect Removal", Proceedings, GUIDE 40, 1975.

[6] B. W. Boeham, "Software Engineering Economics", Prentice-Hall, 1981.
[7] S. D. Chulani, "COQUALMO(COnstructive QUALity MOdel): A Software Defect Density Prediction Model", Project Control for Software Quality, Shaker Publishing, 1999.
[8] S. D. Chulani, "Modeling Software Defect Introduction", USC - Center of Software Engineering, 1998.
[9] B. Boehm, B. Clark, E. Horowitz, R. Modachy, R. Shelby, and C. Westland, "The COCOMO 2.0 Software Cost Estimation Model", USC Center for Software Engineering, 1995.
[10] T. Walton, "Quality Planning for Software Development", Alcatel Networks, 2003.
[11] L. Westfall, "Defect Removal Effectiveness", Linda Westfall, 1996.
[12] E. F. Weller, "Inspection Data Analysis", Software Technology Transition, 2003.
[13] A. Jose, N. K. Anju, and S. K. Pillai, "Closed Loop Defect Removal Model Using Statistical Process Control", Magazines & Journals, American Society for Quality, 2000.
[14] UKSMA, "Quality Standards Defect Measurement Manual", United Kingdom Software Metrics Association, 2000.
[15] A. S. Koch, "Software Quality Data", ASK Process, Inc., 2003.
[16] D. Thakur, "Defect Removal Efficiency", http://www.geeks withblogs.net/dthakur/archive/2004/07/05/7617.aspx.
[17] D. N. Card, "Managing Software Quality with Defects", The Journal of Defense Software Engineering, 2003.
[18] M. Criscione, J. Ferree, and D. Porter, "Predicting Software Errors and Defects", SMASM Conference, 2001.
[19] D. Longstreet, "Test Cases & Defects", http://www. ifpug.com/Articles/defects.htm
[20] L. C. Briand, K. E. Elmam, D. Surmann, I. Wieczork, and K. D. Maxwell, "An Assessment and Comparison of Common Software Cost Estimation Modeling Techniques", International Software Engineering Research Network, Technical Report, ISERN-98-27, 1998.
[21] S. Conte, H. E. Dunsmore and V. Y. Shen, "Software Engineering Metrics and Models", Benjamin/Cummings., 1986.
[22] N. E. Fenton and S. L. Pfleeger, "Software Metrics: A Rigorous and Practical Approach", International Thomson Computer Press, 1997.

저 자 소 개

이 상 운 (Sang-Un, Lee)

1983년~1987년 : 한국항공대학교 항공전자공학과 (학사)
1995년~1997년 : 경상대학교 컴퓨터과학과 (석사)
1998년~2001년 : 경상대학교 컴퓨터과학과 (박사)
2003년 : 강원도립대학 컴퓨터응용과 전임강사
2004년~2007.2 : 국립 원주대학 여성교양과 조교수
2007.3~현재 : 강릉원주대학교 과학기술대학 멀티미디어공학과 부교수
관심분야 : 소프트웨어 프로젝트 관리, 소프트웨어 개발 방법론, 소프트웨어 척도, 분석과 설계 방법론, 소프트웨어 시험 및 품질보증, 소프트웨어 신뢰성, 신경망, 뉴로-퍼지, 그래프 알고리즘
e-mail : sulee@gwnu.ac.kr