□研究論文□

韓國 軍事運營分析 學會誌
第16卷, 第1號, 1990. 6. 30.

소프트웨어의 모듈평가척도와 모듈테스트 시간할당에의 응용+
(An Evaluation Measure of Software Module and Its Application to Allocation of Test Times of Modules)

이창훈*, 김건형**

Abstract

This paper presents an allocation model of test times of software modules. An evaluation measure of modules which is based on the data flow path between modules is used at test phase. An evaluation measure is expressed by the possible path number of data flow for each module, which can be interpreted as an importance of individual module. Three criteria : module test time, module reliabilty, and system reliability is considered in this model. Multi-objective programming, hence, is used to solve this model.

Ⅰ. 서 론

현대 정보화사회에서 컴퓨터가 차지하는 역할과 기능은 점차 증대되고 있다. 컴퓨터 처리시스템이 다양해지고, 정보 시스템속에서 소프트웨어 개발, 유지 및 보수에 대한 체계적인 접근방법의 필요성은 소프트웨어 공학(software engineering)과 소프트웨어 신뢰성(software reliability) 분야를 탄생시켰다.

+ 이 논문은 1988년도 문교부지원 한국학술진흥재단의 자유공모과제 학술연구조성비에 의하여 연구되었음.

* 서울대학교 산업공학과

** *Pennsylvania State University*

소프트웨어와 연관되어 소요되는 비용이 급증되어 왔는데 그 주된 이유는, 소프트웨어는 대부분 공통적 기준과 정확성이 결여되기 쉬운 작업이기 때문이다. 그 결과 생산성 향상과 프로그램의 품질개선이 부진하여 소프트웨어 신뢰성의 저하로 테스트와 유지보수에 많은 비용의 소요로 나타났다[1].

소프트웨어 시스템 개발주기를 단계적 모형으로 나타내면 크게 분석단계, 설계단계, 구현(program and coding) 단계, 시스템테스트단계, 유지보수단계로 나눌수 있다. 그 중에서도 소프트웨어 유지보수단계가 소프트웨어의 분석단계, 설계단계, 구현단계와 시스템테스트단계에 비해 많은 자원이 소요된다. 또한 테스트단계는 소프트웨어 개발작업의 반이상의 자원을 소요한다. 그 이유는 테스트 기법의 정립이 구체화되지 않았고, 테스트 시간과 자원을 적절히 할당, 분배하지 못하여 일정계획의 차질과 비용초과가 발생하기 때문이다. 이런 관점에서 테스트 및 유지보수단계에서 체계적인 기법을 정립하여 적용할 수 있어야만 생산성 개선에 효과를 기대할 수 있다[2].

본 연구는 테스트단계에서 모듈화 소프트웨어 시스템에서 각 모듈의 복잡도와 전체 시스템 구조에서 각 모듈의 자료흐름경로(data flow path)를 이용하여 모듈평가 척도를 설정하는데 있다. 또한 그것을 이용하여 테스트 시간, 모듈의 신뢰성, 전체 시스템의 신뢰성의 기준을 고려한 각 모듈의 테스트시간 할당 모형을 정립하고자 한다.

Ⅱ. 연구 배경

신뢰성에 대한 개념은 하드웨어로 부터 발전되어 1972년 Jelinski와 Moranda[11]에 의해 소프트웨어의 신뢰성에 대한 논의가 시작되었다. Jelinski와 Moranda에 의해 제시된 소프트웨어 신뢰성 모형은 하드웨어의 신뢰성 모형을 소프트웨어에 적용한 것이었다. 이점에 대해 Littlewood는 소프트웨어 신뢰성이론은 소프트웨어가 가지는 특성을 고려해야만 한다고 하였다. 소프트웨어 신뢰성의 가장 일반적인 정의는 "시스템이 주어진 시간동안 주어진 사양(specification)을 따라 성공적으로 실행될 확률"이다. 하드웨어 신뢰성과 소프트웨어 신뢰성의 가장 큰 차이는 소프트웨어의 경우 오류(error)가 시스템내에 존재해도 고장이 발생하지 않는 경우도 있다는 점이다. 따라서 소프트웨어가 계획된 사양대로 실행되지 않았을 때 이를 고장(failure)의 발생으로 보며, 이때 소프트웨어 내에는 고장의 원인이 존재하고, 이를 소프트웨어 오류라고 정의한다.

Goel-Okumoto 모형[9]은 Jelinski-Moranda 모형과 같이 시스템의 초기상태에 포함되어 있는 오류의 갯수를 상수로 보지 않고 확률변수로 간주하여 시스템의 고장률은 시간 t와 시스템내에 남아 있는 평균오류의 갯수에 비례한다고 하였다.

테스트시간 할당 모형으로 Yamada-Osaki 모형과 Kubat-Koch 모형이 있다.

Yamada-Osaki 모형은 소프트웨어 평균비용과 소프트웨어의 신뢰성을 동시에 결정기준으로 하

여 Goel-Okumoto 모형에서 테스트 시간을 구한 모형이다. Kubat-Koch 모형[13]은 소프트웨어가 m개의 모듈로 구성되어 있고 각 모듈에 Jelinski-Moranda 모형의 가정을 적용하였을 때, 각 모듈별 테스트시간 할당에 관한 모형이다.

경로를 이용한 소프트웨어의 신뢰성 모형으로 Downs 모형이 있는데 이것은 테스트 전략의 변화에 대해 각 테스트의 효과를, 경로를 이용하여 평가, 분석한 모형이다.

Yamada-Osaki 모형은 테스트시간 할당 모형에서 테스트 비용과 신뢰성을 고려한 모형으로 의의가 있다. Kubat-Koch 모형에서는 소프트웨어 시스템이 모듈로 구성되어 있을 때 각 모듈을 기준으로 오류의 치명도로 가중치를 두어 테스트 시간을 할당하는 모형이다. 그러나 단일 시스템 연구모형의 범주에서 벗어나지 못했다.

본 논문에서는 모듈 상호간의 관계를 자료흐름 가능 경로의 특성으로 규명하고 자료흐름 그래프의 개념으로 전개하여 모듈의 자료흐름 가능 경로를 구하고, 그것을 이용하여 모듈평가 척도를 설정하였다. 경로를 이용한 소프트웨어 시스템 고장률 유도모형은 소프트웨어 시스템의 각 부분을 통과하는 경로 집합을 도입하여 전체 시스템의 고장률을 설정하였다. 또한 모듈평가 척도를 이용하여 모듈의 고장률과 전체 시스템의 고장률을 유도 할 수 있다. 이것을 의사결정의 기준과 조건의 변화를 정해진 제약조건하의 다목적 함수 계획법으로 모형화하고 비용을 고려한 테스트시간 할당 모형으로 제시하였다.

Ⅲ. 소프트웨어 시스템 모듈평가 척도

Ⅲ-1. 모듈화 소프트웨어 시스템

1. 조 건

모듈화 소프트웨어 시스템은 시스템의 구축, 프로그램의 설계과정에서 구조화 설계가 시행되었다.[1]

구조화 설계는 소프트웨어 시스템의 모듈화이다. 모듈의 크기에 대한 정의는 컴파일(compile)의 단위로서 크기는 100이상의 문장(statement)으로 한다.

2. 구조도

앞에서 제시한 모듈화 소프트웨어시스템의 조건으로부터 다음 그림3-1과 같은 구조도(structured diagram)가 제시된다.

자료흐름이 구조도에서는 입력, 처리, 출력시스템으로 변환되는 입력, 처리, 출력부분으로 구성되어있다. 시스템에서 세개의 주요 서브시스템 사이의 경계는 자료흐름에서 가장 추상적인 입력자료와 가장 추상적인 출력자료가 있는 지점을 결정함으로써 밝혀진다.

3. 자료흐름 그래프(data flow graph)

그림 3-1에서 제시된 구조에서 각 노드가 모듈과 대응되는 방향그래프(directed graph)로 표현된다. G=(N, E)는 다음과 그림 3-2와 같이 표시된다. 여기에서 E(i,j)는 모듈 i와 j사이

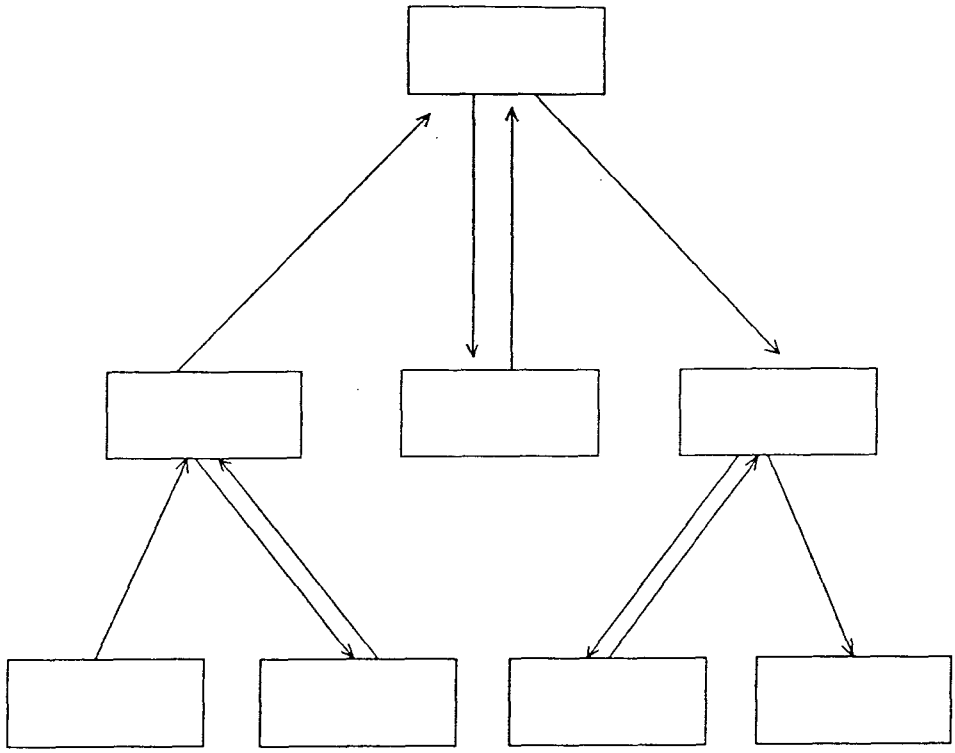


그림3-1 모듈화 소프트웨어 시스템의 계층적 구조

의 자료흐름 빈도수를 나타내고 e_{ij}는 모듈 i에서 j로의 자료흐름 경로를 나타낸다. CN(i)는 모듈 i의 cyclomatic number를 표시한다. 이것은 그래프내의 일차독립인 경로수이며 구조화된 프로그램안에서 cyclomatic number는 술어의 수에 1을 더한 것과 같고, 분기 결정구조 즉 Decisions, ANDs, ORs, NOTs문장수의 총합 더하기 1을 한 것과 같다.

4. 가 정

모듈화 소프트웨어 시스템의 기본적 가정은 다음과 같다.

(1) 시스템은 m개의 모듈로 구성된다.

(2) 각 모듈의 초기 오류(error) 갯수는 Ni(i=0 to m)이다.

(3) 모듈의 고장률(failure rate)은 모듈내에 남아있는 오류의 수에 비례한다.

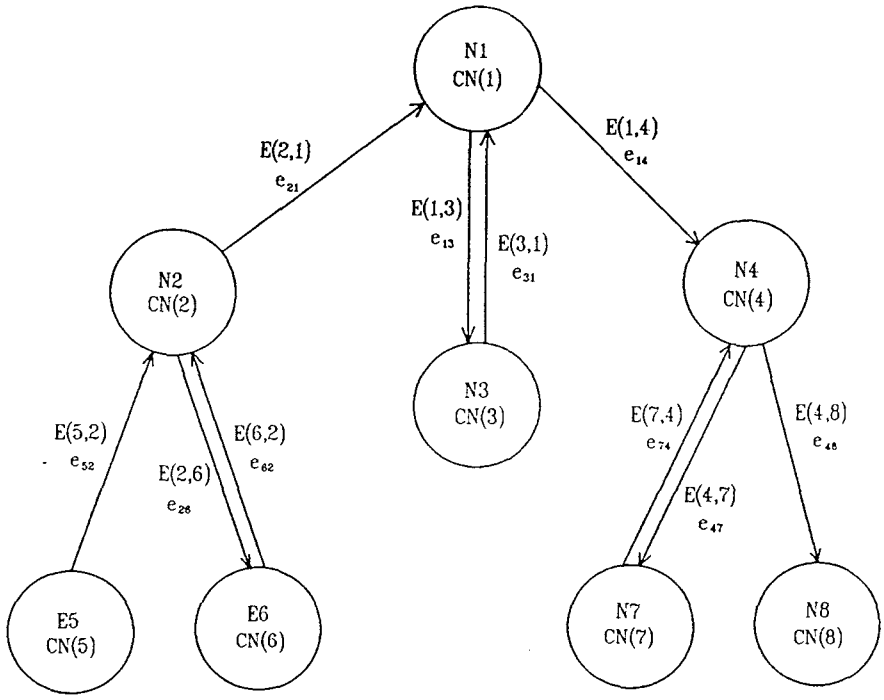
(4) 오류는 각각 같은 비율로 모듈의 고장에 영향을 미친다.

(5) 오류는 1개씩 발견된다.

(6) 발견된 오류는 즉시 완전하게 고쳐진다.

(7) 고장시간은 지수분포(exponential distribution)을 따른다.

(8) 오류들은 상호 독립이다.



E(i,j) : 모듈 i에서 j까지의 자료흐름 빈도 경로수

e_{ij} : 모듈 i에서 j로의 자료흐름 경로

CN(i) : 모듈 i의 cyclomatic Number

그림3-2 자료 흐름 그래프(data flow graph)

Ⅲ-2 모듈평가 척도 설정

1. 측정 변수 및 기호

U(G) : G의 unisignant matrix

Ui(G) : 노드 i를 제외한 G의 unisignant matrix

det U(G) : U(G)의 determinant

det Ui(G) : Ui(G)의 determinant

N : 자료흐름 그래프에 나타난 노드의 index set

β : Ui(G)에서 생성된 각 경로에서 e_{ik}, k ∈ PN(i, r) 첨가시 생성되는 cycle의 수

PN(i, j) : Ui(G)에 의해 생성된 경로집합중 j번째 경로에 속하는 노드의 index set

|PN(i, j)| : 집합 PN(i, j)의 원소 갯수

IDP(i) : 노드 i가 제거되지 않았을 때 i를 경유하나 i를 종착노드로 하지 않

는 자료흐름 경로의 수

IDPC(i) : 노드 i 제거시 i를 종착노드로 하는 자료흐름 경로 수

TIDP(i) : 경로 i 제거시 G의 자료흐름 경로수

F(i, j) = CN(i) E(i, j)

CDR(i) : 모듈 i의 치명도(criticality)

TDFP : 총 자료흐름 경로수

2. 자료흐름 경로(data flow path)

자료흐름 그래프 G = (N, E)에 제시된 cyclomatic number와 각 모듈의 E(i, j)와 e_{ij}를 이용하여 자료흐름의 가능경로를 경로집합으로 나타낸다.

노드 m개의 방향그래프 G의 unisignant matrix U(G)는 다음 식(3-1)과 같이 표시된다[5].

$$U(G) = \begin{bmatrix} \sum_{k=2}^{m} e_{1k} & -e_{12} & -e_{13} & \cdots & -e_{1m} \\ -e_{21} & \sum_{\substack{k=1 \\ k\neq 2}} e_{2k} & -e_{23} & \cdots\cdots & -e_{2m} \\ \vdots & & & & \\ -e_{(m-1)} & -e_{(m-1)2} & \cdots & \sum_{\substack{R=1 \\ R\neq m-1}}^{m} e_{(m-1)R} & \end{bmatrix} \quad (3\text{-}1)$$

det U(G)는 모든 노드를 통과하는 가능한 경로의 집합과 각 경로를 구성하는 e_{ij}를 생성한다. det Ui(G)는 G의 모든 가능 경로중 노드 i를 시발점으로 하는 모든 경로와 노드 i를 통과하는 경로를 제외한 모든 경로를 구한다. 단, 노드 i를 종착점으로 하는 경로는 포함된다.

det U(G)와 det Ui(G)를 경로집합 PN과 연결하여 식을 나타내면 다음식(3-2), 식(3-3)과 같다.

$$\det U(G) = \sum_{i=1}^{m-1} e_{im} U_{i1}(G) = \sum_{i=1}^{|PN(m)|} \{ \prod_{i,\, j \in PN(m,\, l)} e_{ij} \} \quad 단,\ i \neq j \quad (3\text{-}2)$$

$$\det U_k(G) = \sum_{i=1}^{|PN(k,\, l)|} \{ \prod e_{ij} \} \quad 단,\ i \neq j,\ i \neq k \quad (3\text{-}3)$$

3. 자료흐름 경로수(data flow path number)

자료흐름 그래프 G = (N, E)의 가능한 총 자료흐름 경로수 TDFP는 다음 식(3-4)와 같이 나타난다.

$$TDFP = \sum_{i=1}^{|PN(m)|} \{ \prod_{i,\, j \in PN(m,\, l)} F(i, j) \} \, CN(P) \quad (3\text{-}4)$$

단, P : 종착 노드

노드 i가 제거되지 않았을 때, i를 경유하나 i를 종착노드로 하지 않는 자료흐름 경로의 수 IDP(k)는 다음 식(3-5)와 같이 나타난다.

$$IDP(k) = \sum_{i=1}^{|PN(k,\, l)|} \{ \prod_{i \in PN(k,\, l)} F(i, j) \} \quad (3\text{-}5)$$

단, k는 PN(k, l)에 포함되지 않음.

노드 k가 제거되었을 때 k를 종착노드로 하는 자료흐름 가능 경로수 IDPC(k)는 다음 식(3-6)과 같이 나타난다.

$$IDPC(k) = \sum_{i=1}^{i \in PN(k,\, l)} \{(-1)^{\beta} \prod_{i,\, j \in PN(k,\, l)} F(i, j) \} \, CN(R) \quad (3\text{-}6)$$

단, $k \in PN(k, l)$

β : e_{ij} 첨가시 생성되는 Cycle의 수

그리고 TIDP(k)는 다음 식(3-7)과 같이 나타난다.

$$TIDP(k) = IDP(k) + IDPC(k) \quad (3\text{-}7)$$

자료흐름 그래프 G=(N, E)의 총 자료흐름 가능 경로수에서 노드 k가 영향을 미치는 자료흐름 가능 경로수를 ODP(k)로 표시하면 다음 식(3-8)과 같이 나타난다.

$$ODP(k) = TDFP - TIDP(k) \quad (3\text{-}8)$$

4. 모듈평가 척도 계산방법

총 자료흐름 경로수 TDFP와 각 노드의 자료흐름 가능 경로수 ODP를 이용하여 모듈평가 척도 EM과 전체 자료흐름 경로에 대한 각 노드의 치명도 CDR을 구한다. 관계식은 다음 식(3-9), 식(3-10), 식(3-11)로 나타난다.

$$TODP = \sum_{i=1}^{m-1} ODP(i) \quad (3\text{-}9)$$

$$EM(i) = 1 - \exp(-ODP(i)/TODP) \quad (3\text{-}10)$$

$$CDR(i) = ODP(i)/TDFP \quad (3\text{-}11)$$

여기에서 모듈평가 척도와 모듈의 치명도는 빈도(frequency)와 가능성(possibility)의 개념으로 유도되었다. 모듈화 소프트웨어 시스템의 가정에서 각 모듈의 고장시간은 지수분포를 따른다고 가정하였다.

따라서 모듈평가 척도가 식(3-10)과 같이 주어진다.

Ⅳ. 테스트시간 할당 모형

Ⅳ-1. 모듈평가 척도를 이용한 모듈의 고장률 유도

테스트시간 할당 모형을 정립하기 위해 모듈평가 척도 EM과 모듈의 치명도 CDR를 이용하여 각 모듈의 고장률과 m개의 모듈로 구성된 소프트웨어 시스템의 고장률을 유도하였다.

Downs모형으로부터 고장률은 다음 식(4-1)과 같이 나타난다[8].

$$\lambda = -Nr \ Log \ (1-C/M) \quad (4\text{-}1)$$

단, N : 소프트웨어 시스템의 초기 오류 갯수

r : 단위 시간당 수행 경로의 수

C : 1개의 오류로 부터 영향을 받는 경로의 수

M : 시스템 전체의 경로수

모듈 i를 통과하는 자료흐름 가능 경로수를 Ci라 하면 모듈 i의 고장률 λi는 다음 식(4-2)와 같이 나타난다.

$$\lambda_i = -N_i r \ Log(1 - C_i/M) \quad (4\text{-}2)$$

단, C_i : 모듈 i를 통과하는 자료흐름 가능 경로수

식(4-2)에서 Ci의 변화에 따른 λi의 결정을 다음 그림 4-1과 같이 나타낼 수 있다.

즉, 전체 소프트웨어 시스템에서 모듈 i를 경유하는 경로수가 많을수록 모듈의 고장률도 증가하는 것으로 나타난다.

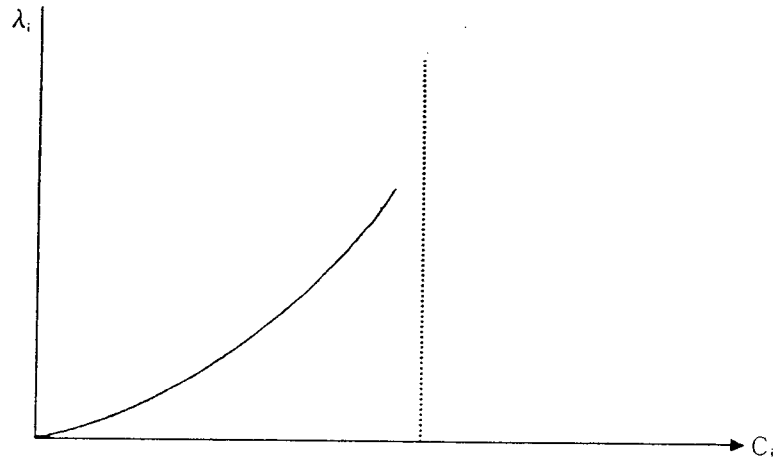


그림4-1 자료흐름 가능경로수의 변화에 대한 고장률 결정

Downs 모형으로부터 m개의 모듈로 구성된 소프트웨어의 고장률 λ_r는 다음 식(4-3)과 같이 표현된다.

$$\lambda_r = -r\,\mathrm{Log}\{\sum_{i=1}^{m} P_i (1 - C_i/M_i)^{N_i}\} \qquad (4\text{-}3)$$

단, P_i : 모듈 i가 테스트될 확률

N_i : 모듈 i의 초기 오류 갯수

식(4-3)에 제시된 λ_r에서 각 모듈이 테스트될 확률 P_i를 유도하였다.

여기서 사건 A_i를 모듈 i를 통과하는 자료흐름 경로수가 P_o^i를 초과하는 경우로 정의하면, A_i는 다음 식(4-4)와 같이 나타낼 수 있다.

$$A_i = \{C_i | C_i > P_o^i\} \qquad (4\text{-}4)$$

단, $P_o^i = \mathrm{ODP}(i)$

식(3-10)과 참고문헌 [12]를 참조하여 사건 A_i가 일어날 확률 $P(A_i)$를 다음 식(4-5)와 같이 유도할 수 있다.

$$P(A_i) = 1 - (1 + b_i/P_o^i)^{-2} \qquad (4\text{-}5)$$

식(4-5)에서 b_i는 오류가 포함된 전체 경로중에서 모듈 i를 통과하는 경로의 고장률을 나타내며, 식(4-2)를 이용하여 식(4-6)과 (4-7)로 표현된다.

$$b_i/P_o^i = \lambda_i/Nr \qquad (4\text{-}6)$$

$$b_i = (\lambda_i/Nr)P_o^i \qquad (4\text{-}7)$$

식(4-3)에서 모듈 i가 테스트될 확률 P_i를 식(4-5)의 $P(A_i)$에서 유도하는 과정은 다음 그림 4-2와 같이 나타낼 수 있다.

그림 4-2에서 $P(A_i)$는 사건 A_i가 일어날 확률 즉, 모듈 i를 통과하는 경로의 수가 P_o^i를 초과할 확률을 말한다. 또한, 모듈 i를 통과하는

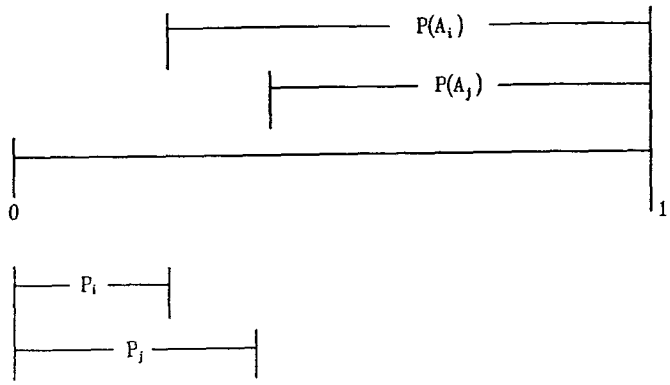


그림4-2 P_i와 $P(A_i)$의 관계식 유도

경로의 수가 P_o^i를 초과하면, P_i는 임의의 경로가 모듈 i를 통과할 확률을 나타낸다.

Ⅳ-2. 테스트 시간할당모형

앞의 식(4-2)와 (4-3)을 다음 식(4-8)을 이용하여 식(4-9), 식(4-10)으로 나타낼 수 있다.

$$EM(i) = C_i/M_i = 1 - Q_i \quad (4\text{-}8)$$

$$\lambda = -N_i r \operatorname{Log} Q_i \quad (4\text{-}9)$$

$$\lambda_r = -r \operatorname{Log}\{\sum_{i=1}^{m} P_i Q_i^{N_i}\} \quad (4\text{-}10)$$

모듈화 소프트웨어 시스템이 Jelinski-Moranda 모형의 가정을 따른다는 가정으로부터 다음 식(4-11), (4-12)가 유도된다[14].

$$N_i = K_i N_i / (1 - e^{-r\lambda_i t_i}) \quad (4\text{-}11)$$

$$\lambda_i = \operatorname{Log}(1 - K_i)/T_i \quad (4\text{-}12)$$

단, K_i : 테스트 시간 T_i를 할당하는 경우, 검출해 내고자하는 오류의 비율($0 < K_i < 1$)

모형의 설정에 따르는 기준으로 전체 테스트 시간 T와 각 모듈의 신뢰성 R_o^i, 전체 시스템 신뢰성 R_o가 제시되었을 때 다음 식(4-13), (4-14), (4-15)로 나타난다.

$$\sum_{i=1}^{m} T_i \le T \quad (4\text{-}13)$$

$$-N_i r \operatorname{Log} Q_i = \operatorname{Log}(1 - K_i)/T_i \ge \overline{R_o^i} \quad (4\text{-}14)$$

$$-r \operatorname{Log}(\sum P_i Q_i^{N_i}) \ge \overline{R_o} \quad (4\text{-}15)$$

단, T_i : 모듈 i의 테스트 할당시간

$\overline{R_0^i} = 1/R_0^i$

$\overline{R_o} = 1/R_o$

다음 식(4-16)을 이용하여 식(4-14)는 식(4-17)로 표현될 수 있다.

$$E_i = \operatorname{Log}(1 - K_i)/r \operatorname{Log} Q_i \quad (4\text{-}16)$$

$$N_i = E_i/t_i \ge R_o^i \quad (4\text{-}17)$$

식(4-18), 식(4-19)를 이용하여 식(4-15)는 다음 식(4-20)으로 표현될 수 있다.

$$R = Exp(-R_o/r) \quad (4\text{-}18)$$

$$F_i(T_i) = P_i Q_i^{1/T_i} \quad (4\text{-}19)$$

$$R(T) = \sum_{i=0}^{m} F_i(T_i/E_i) \leq R \quad (4\text{-}20)$$

다목적 함수 모형으로 정립하기 위해 모형의 설정에 따르는 기준으로 테스트 할당시간, 모듈의 신뢰성, 전체 시스템의 신뢰성이 제시되었다.

기준의 우선순위(priority) 배열을 여러가지 조건에 따라 변화시킬수 있지만, 본 연구에서는 테스트 할당시간, 각 모듈의 신뢰성, 전체 시스템의 신뢰성 순으로 배열하였다. 이것은 모형에서 다음 식(4−21)과 같이 표현되고 사전식 우선순위 최소화(lexicographically minimize)로 정의된다[10].

사전식 우선순위 최소화 : $\{\rho T, \sum_{i=1}^{m} n_i, n_R\}$

식(4−21)에 나타난 변수는 식(4−13), (4−17), (4−20)에 부가된 deviation변수를 나타낸다.

각 deviation 변수가 구속되는 제약조건(constraint)은 다음 식(4−22), (4−23), (4−24)로 나타난다.

제약조건(constraint) :

$$\sum_{i=1}^{m} T_i + n_T - \rho T = T \quad (4\text{-}22)$$

$$-E_i/T_i + n_i - \rho_i = R_0^{-i} \quad (4\text{-}23)$$

$$\sum_{i=1}^{m} F_i(T_i/E_i) + n_R - \rho R = \overline{R} \quad (4\text{-}24)$$

Ⅳ-3. 비용을 고려한 테스트시간 할당 모형

모형의 설정에 따르는 기준을 신뢰성, 테스트 시간, 비용으로한 모형을 제시하였다.

전체 소프트웨어 시스템의 신뢰성을 기준으로 할당된 테스트시간에 따라 발생하는 소요비용을 목적함수로 하여 (4−25)와 같이 나타내었다.

T시간의 테스트 후 비용의 최소화 :

$$C(T) = C_3 N \exp(-T\lambda_T) + C_4 T$$
$$= C_3 N \exp\{-T, \mathrm{Log}\{\sum P_i (1 - C_i/M)^{N}\}\}$$
$$+ C_4 T \quad (4\text{-}25)$$

또한, 각 모듈의 신뢰성과 테스트 할당시간을 목적함수의 제약조건으로 식(4−26), (4−27)로 나타난다.

제약조건(constraint) :

$$\sum_{i=1}^{m} \{C_4 T_i + C_3 N_i \exp(-T_i \lambda_i)\} \leq C_0 \quad (4\text{-}26)$$

$$\sum_{i=1}^{m} T_i = T \quad (4\text{-}27)$$

단, T_i : 모듈 i의 테스트 할당 시간

C_0 : 주어진 constraint

C_3 : 테스트 종료후 발생하는 오류의 평균 수리비용과 평균손해비용의 합

C_4 : 단위 시간당 평균테스트 비용

N : 시스템의 초기 오류 갯수

식(4−25)에서 나타난 $N\exp(-T\lambda_T)$는 테스트 시간 T를 할당하는 경우 예상되는 오류의 평균 잔존갯수를 의미한다.

Ⅴ. 결론 및 토의 사항

본 연구에서는 지금까지의 단일모듈

Black-Box 모형에서 다중모듈의 신뢰성 모형으로 확장 정립하였다. 그리고, 이러한 모듈의 평가척도로서는 소프트웨어 시스템의 자료흐름 가능 경로수를 사용하였다.

또한, 자료흐름 가능 경로수와 시스템의 신뢰성 관계를 이용하여 모듈과 전체 시스템의 신뢰성을 기준으로 한 테스트시간 할당 모형을 정립하였다. 더불어, 테스트 기준과 조건의 변화에 대하여 다목적함수 계획 모형과 비용을 고려한 모형으로 발전시켰다.

다음과 같은 결과를 위의 연구 내용으로부터 도출하였다.

1. 각 모듈을 경유하는 자료흐름 경로수가 많을수록 모듈의 고장률도 증가하는 것으로 나타났다.

2. 같은 테스트 시간을 할당하는 경우 자료흐름 가능 경로수가 많은 모듈이 신뢰도 증가가 작은 것으로 나타났다. 따라서 같은 크기의 신뢰도 증가를 필요로 할 경우에 자료흐름 가능 경로수가 많은 모듈에 많은 테스트 시간을 할당해야 하는 것으로 나타났다.

본 연구에서는 전체 소프트웨어 시스템을 모듈이라는 부분 시스템으로 범위를 줄여 적용하였다.

모듈의 소프트웨어 시스템에서의 상호 역할 기능과 관계등을 고려하여 그것을 이용하여 모듈의 자료 흐름 경로수로 나타내었다.

그것을 이용하여 모듈평가 척도를 설정하고 모듈의 고장률을 고려한 전체 소프트웨어 시스템의 신뢰성을 나타내었다. 즉, 모듈의 범위에서 전체 시스템의 신뢰성을 나타내었다.

추후 연구방향으로 본 연구에서 고려되었던 자료흐름 경로외에 소프트웨어 시스템의 특성을 일반적으로 표출하는 특성기준을 고려하여 전개시킬 수 있다. 본 연구에서는 모듈평가 척도를 자료흐름 경로수로서 유도하였다. 그러나, 그 외 일반적 특성의 기준을 부가하여 각 모듈의 통제(control) 흐름에 관한 척도, 자료와 통제흐름의 상호 관련 척도를 도입하고 새로운 테스트시간 할당 모형을 정립할 수 있을 것이다.

참 고 문 헌

1. 이기식, 정왕호, 소프트웨어 생산기술, 정익사, 1984
2. 이동필, 최병필, 소프트웨어 공학, 상조사, 1986
3. Athur, L., *Measuring Programmer Productivity and Software Quality*, John Wiley & Sons, New York, 1985.
4. Baker, A. and Zweben, S., "Comparison of Measures of Control Flow Complexity," *IEEE Trans. on Software Engineering*, Vol. SE-6, pp. 506~511, 1980.

5. Chen, W., *Applied Graph Theory*, North-Holland Publishing Company, 1980.
6. Chen, E., "Program Complexity and Program Productivity," *IEEE Trans. on Software Engineering*, Vol. SE-4, pp. 187~194, 1978.
7. Deo, N., *Graph Theory with Application to Engineering and Compuler Science*, Prentice Hall. Inc, 1974.
8. Downs, T., "An Approach to Modeling of Software Testing with Some Applications," *IEEE Trans. on Software Engineering*, Vol. SE-11, pp. 375~386, 1985.
9. Goel, A. and Okumoto, K., "Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measure," *IEEE Trans. on Reliability*, Vol. R-28, No. 3, pp. 206-211, 1979.
10. Ignizio, J., *Linear Programming in Single and Multiple-Objective System*, Lexington, Mass : Heath, 1976
11. Jelinski and Moranda, "Software Reliability Research," in W. Freiberger Ed., *Statistical Computer Performance Evaluation*, Academic Press, New York, 1972.
12. Kandel, A., "Comments on the Minimization of Fuzzy Functions," *IEEE Trans. on Compulers*, Vol. C-22, pp. 217, 1973.
13. Kubat, P. and Koch, H., "Managing Test-Procedures to Achieve Reliable Software," *IEEE Trans. on Reliability*, Vol. R-32, pp. 299~303, 1983.
14. _____, "Programatic Testing Protocols to Measure Software Reliability," *IEEE Trans. on Reliability*, Vol. R-32, pp. 338~341, 1983.
15. McCabe, T., "A Complexity Measure," *IEEE Trans. on Sofrware Engineering*, Vol. SE-6, pp. 489~500, 1980.
16. Myers, G., *Reliable Sofrware through Composite Design*, Litton Educational Publishing. Inc., 1975.
17. Yamada, S. and Osaki, S., "Cost-Reliability Optimal Release Policies for Software System," *IEEE Trans. on Reliability*, Vol. R-34, pp. 422~424, 1985.