공수 예측 모델 유도를 위한 자료 흐름도의 실험적 평가

김 명 옥† 백 청 호†† 양 해 술†††

요 약

소프트웨어 개발자와 사용자에게 직면한 가장 중요한 문제는 프로그래밍 시스템의 크기와 개발 노력의 예측이라고 할 수 있다. 본 논문에서는 자료 흐름도, 자료 사전, 소단위 명세서로 구성된 구조화 명세서에 관한 각각의 특성을 정의하고, 구조화 명세서의 정량적인 평가 요소를 프로그램 메트릭스에 적용하였다. 또한 구조화 명세서를 구성하는 자료 흐름도에 관해서는 정량적인 평가 실험을 하였다. 그 결과 상위 공정의 분석 단계에서의 산출물에 의한 하위 공정에 관한 공수 예측 모델을 제안한다.

Experimental Estimation of Data Flow Diagram for Man/Month Prediction Model Derivation

Myoung Ok Kim†, Cheong Ho Back†† and Hae Sool Yang†††

ABSTRACT

One of the most important problems faced by software developers and users is the prediction of the size of programming system and its development effort. This article define the identical characteristics for structured specification which is consisted of Data Flow Diagram, Data Dictionary and Mini Specification and apply quantitative estimation factor of structured specification to program code metrics. Moreover, concerning DFD which is made up of component element of structured specification executed quantitative estimation experiment. In the result, we propose man/month prediction model of lower progression with production on analysis phase of upper progression.

1. 서 론

다른 공학과는 달리 소프트웨어 공학에서는 시스템의 주요 기능을 정확히 파악하지 못한 상태에서 설계나 제작에 착수해야 하는 경우가 많이 발생한다. 이와같이 요구 사항이 분명치 않은 경우에는 요구 분석 및 설계 등의 개발 초기 단계의 잘못으로 완성된 소프트웨어가 사용자의 요구에 합당하지 않은 경우가 발생하여 사용자에게 인도된 후에도 많은 수정이 필요하거나 아예 인도되기 전에 폐기되는 사례가 종종 발생되고 있다[1].

† 종신회원: 한컴데이타(주) 기술연구소 연구원
†† 종신회원: 강원대학교 전자계산학과 부교수
††† 종신회원: 강원대학교 전자계산학과 교수

논문접수: 1994년 12월 1일, 심사완료: 1995년 1월 14일

따라서 소프트웨어 개발은 초기 단계에서 각종 지식을 바탕으로 여러가지 사항을 예측해서 그에 필요한 수단을 강구해 두는 것이 효율적인 소프트웨어 개발에 필수적인 사항이라고 할 수 있다[14]. 또한 소프트웨어 개발주기의 상위공정 단계인 요구 분석과 설계 단계에서 하위공정에 관한 공수의 예측 및 하위공정에서 생성되는 프로덕트의 품질과 규모를 예측하는 것은 소프트웨어 개발, 관리에 관한 스케줄링, 자원 할당을 처리하는데 있어서 대단히 중요한 작업이라고 할 수 있다. 따라서 상위공정 단계에서 하위공정 단계의 품질규모를 예측하기 위해서 상위공정의 결과물을 분석하고, 규모에 영향을 줄 수 있는 정량적인 요인을 분석하는 것이 바람직하다.

본 연구에서는 요구분석 단계의 산출물인 자료

흐름도(Data Flow Diagram), 자료 사전(Data Dictionary), 소단위 명세서(Mini Specification)로 구성된 구조화 명세서를 통해서 시스템의 규모와 공수를 예측할 수 있는 방법을 제시하며, 공수 예측 모델을 유도하기 위해 구조화 명세서에 포함된 자료 흐름도의 실험적인 평가를 하였다.

본 연구에서는 구조화 명세를 구성하는 3종류의 문서 전체에 대해서 동일 특성을 갖는 각각의 요소를 추출하였으며, 명세서 자체에서 추출된 오퍼레이터와 오퍼랜드를 정의하여 기존의 평가 척도에 적용하였다. 또한 각각의 명세서에서 하위공정의 공수에 영향을 주는 요인을 파악하여 구조화 명세서 전체를 통한 공수 예측 방법을 제시하였다. 그리고 공수 예측 모델을 유도하기 위한 평가 실험에서 자료 흐름도에 관해서 하위공정의 공수에 영향을 주는 특성을 선정하였고, 자료 흐름도를 의사코드로 변환하는데 걸리는 시간을 계측하였다. 그리고 각 실험의 조사 결과에 대한 분석은 통계 패키지인 SAS(Statistical Analysis System)를 통한 분산분석법을 사용하여 각 요인에 관한 영향도 측정을 통해서 하위공정에 대한 공수 예측 모델을 유도하였다.

2. 구조화 분석법

구조화 분석법은 효율적인 시스템 분석 명세서를 작성하기 위해서 현재 널리 사용되고 있는 시스템 개발 방법 중의 하나로 도형 중심의 분석도구를 이용하며, 시스템 요구명세의 구조화된 모델을 도출하기 위한 기법으로 주로 사무처리 시스템의 모델화에 이용된다. 그리고 구조화 분석에 사용되는 분석 도구로 자료 흐름도, 자료 사전 및 소단위 명세서라 부르는 세종류의 문서 도구를 사용한다[2, 4]. 이와 같은 구조화 분석 기법에 의한 산출물 전체를 구조화 명세서(structured specification)라고 부른다[5, 6].

본 연구에서는 구조화 분석에서 사용되는 3종류의 구조화 명세서를 H라고 표현하고, 각각의 명세를 DFD, MS, DD로 표기한다.

- DFD: Data Flow Diagram D_L
- DD: Data Dictionary Dd_L
- MS: Mini Specification Ms_L

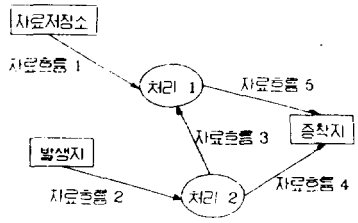
단, L은 단계(level)로서 $n_1n_2 \cdots n_i$(각 n_i는 양수)의 형을 갖는다.

H를 구성하는 DFD, DD 및 MS에서는 그들간의 관계와 복잡한 시스템을 이해하기 쉽도록 표현하기 위해 단계화를 사용한다. 단계화된 DFD의 최상위 단계는 배경도(CD: Context Diagram)로 개발 대상 업무의 상세한 부분은 분할되어 있지 않으나 분석 대상 시스템의 범위 설정과 시스템 외부와의 인터페이스 및 분석 대상 시스템의 개요 파악을 위해 사용된다.

2.1 구조화 명세의 종류

2.1.1 자료 흐름도

자료 흐름도는 시스템을 망형으로 표현한 것으로서 시스템을 그 구성 요소로 나타내고 아울러 이 구성 요소들 사이의 모든 연결 관계도 표현한다. 바꾸어 말하면 모델을 설정하고 기능적으로 분할하여 명세서를 작성하는데 이용된다. 자료 흐름도는 그림을 통해 분할되고 자료의 흐름을 강조하는 반면 제어(control)의 흐름은 표시하지 않는 특성을 갖고 있으며, (그림 1)과 같이 자료흐름(data flow), 처리(process), 자료 저장소(data store), 자료 발생지(source)와 종착지(sink)로 구성되어 있다. 본 논문에서는 자료 발생지와 종착지를 연결점이라 하고, '*'로 나타낸다.



(그림 1) 자료 흐름도
(Fig. 1) Data Flow Diagram

2.1.2 자료 사전

자료 사전이란 자료 흐름도에서 선언된 연관 관계들을 정의해 놓은 집합체로서 자료 저장과 자료 흐름도상에 나타난 모든 자료 흐름에 대한 자료 정의의 집합이다. 이것은 자료 흐름과 자료 저장소를 그 구성 요소로 표시하는데 구성 요소 그 자체가 자료 흐름일 수도 있고, 또는 자료 원소(data element)일 수도 있다. 데이타는 더 이상의 요소로 분해할 수 없는 데이타로 최소 단위 데이타(primitive data)와 여러 종류의 데이타 항목으로 이루어진 합성 데이타(composite data)의 두가지로 분류될 수 있다. 데이타 사전은 〈표 1〉과 같은 기호 연산자를 사용하여 최소 단위 데이타와 합성 데이타를 정의한다.

〈표 1〉 자료 사전의 기호 연산자
〈Table 1〉 Symbol Operator of Data Dictionary

기호	의 미
=	자료항목 정의에 사용된다. (Equivalence)
+	복합적인 자료요소의 구성을 나타낸다. (Sequence)
[|]	내용 중의 택일을 의미한다. (Selection)
{}	내용을 1이상 반복한다. (Iteration)
n{}	내용을 n이상 반복한다.
{}m	내용을 1부터 m까지 반복한다.
n{}m	내용을 n부터 m까지 반복한다.
( )	생략 가능함을 나타낸다. (Option)
**	주석을 나타낸다. (Remark)

2.1.3 소단위 명세서

소단위 명세서는 자료 흐름도에 나타난 모든 최소 단위 처리(primitive process)의 입력 자료 흐름을 출력 자료 흐름으로 변환하는 과정을 기술하는 도구이다. 소단위 명세서는 매우 간단한 단어들로 처리과정과 관련된 모든 주요활동을 기술하는 것으로서 단계화된 자료 흐름도의 최하위 처리의 입력 자료 흐름을 출력 자료 흐름으로 변환하는 과정을 명세화해 놓은 문서이다.

소단위 명세서는 형식화된 기법을 이용하기 위해서 처리에 대한 입력 데이타로부터 출력 데이타가 생성되는 절차와 조건분기 등의 제어구조를 고려해야 한다. 의사코드에 의한 기술은 상기 2가지를 여러가지 구체적인 표현으로 나타내기 쉬우나 자연어에 의한 기술에서는 특히 제어구조를 파악하는 것이 곤란하다. 본 연구에서는 자연어에 의한 기술을 형식적인 모델로 변환하기 위해서 마침표(period)에 의해 끝나는 문장을 단위로서 다룬다. 각 문은 어떤 데이타의 집합 d_a에 의해 생성되는 데이타의 집합 d_b로 구성되어 있으며 단문 S_i에서 포함된 동사(구)를 v라고 한다면, (식 1)과 같이 세항목으로 나타낸다.

$$S_i = (d_a,\ d_b,\ v) \qquad (1)$$

이것은 입력 데이타의 d_a에 처리 v를 행하여 출력 데이타의 집합 d_b를 생성하는 처리를 표현한 것이다. 중문의 경우에는 중문을 구성하는 단문을 세항목으로 구성된 여러 개의 문장으로 표현하고, 중문에 나타난 순서로 나열한다. 이 절차에 의해 임의의 중문에서 n개의 단문 S_1, S_2, …, S_n의 순서로 구성된 문장을 S_1; S_2; …, S_n; 과 같이 표기한다.

그리고 조건문에서는 조건의 판정 대상으로 된 데이타 집합 d_a와 조건마다 실행되는 n 개의 처리 집합 $\{S_1, S_2, \cdots, S_n\}$으로 이루어진다. 이와 같은 조건문 S_i를 (식 2)와 같이 표기하며, 여기에서 각 S_i(단, i=1, 2, …, n)는 단문 및 조건문의 순차적인 전개를 나타낸다.

$$\begin{aligned} S_i = d_i\{\ & S_1; \\ & S_2; \\ & \ \vdots \\ & S_n; \\ & \}; \end{aligned} \qquad (2)$$

이상의 정의에 의해 임의의 자연어에 관한 소단위 명세서의 기술을 S_a, S_i라고 하는 2종류의 구조에 의해 형식적인 기술로 변환할 수 있다.

2.2 구조화 명세의 특성

2.2.1 구조화 명세와 프로그램 코드의 대응 관계

구조화 명세는 개발하려고 하는 시스템의 모델이라고 할 수 있기 때문에 프로그램 코드의 구조

와 비교 가능한 부분이 많이 존재한다. C 언어 코드와의 대응을 통해서 구조화 명세의 구조를 살펴보면, C에 있어서 main 함수는 프로그램 코드에 관한 시스템의 범위와 외부의 인터페이스를 기술하고 있다. 또한 프로그램 코드 내의 모든 함수는 main 함수로부터 함수호출을 한 순서대로 도달할 수 있다. 따라서 구조화 명세에서 main 함수에 해당하는 것이 배경도인 CD라고 할 수 있다. 그리고 프로그램 코드에서는 main 함수가 함수 a를 호출하고, a가 또 다른 함수인 b를 호출할 경우에는 함수의 친자관계가 존재한다.

구조화 명세에서 이러한 관계는 상위 단계의 DFD와 하위 단계의 DFD의 친자관계에 해당하며, 임의의 DFDd_i은 별도의 DFD(또는 CD)d_0의 어떤 하나의 처리를 보다 상세화한 것에 해당한다. 또한 프로그램 코드에서는 다른 복수의 함수를 호출하지 않는 최하층의 함수가 존재한다. 이것은 코드에 있어서 가장 기본적인 기능을 구현하고 있는 것이므로 구조화 명세에서는 기본 기능을 기술하고 있는 MS에 대응시킬 수 있다. 그리고 코드에 있어서 데이타의 종류 및 구조 등은 헤더화일과 각 함수의 변수 선언부에 기술되어 있으므로 구조화 명세에 해당하는 것은 DD라고 할 수 있다.

2.2.2 특성의 정의

구조화 명세와 프로그램 코드를 비교함으로써 프로그램의 특성과 유사한 것을 구조화 명세에 정의하는 것이 가능하다고 본다. 본 절에서는 프로그램 코드의 특성으로서 Halstead의 Software Science[8]에서 착안한 오퍼레이터와 오퍼랜드의 개념을 구조화 명세의 각 문서의 특성에 대응시키며, 〈표 2〉에 구조화 명세서의 오퍼레이터와 오퍼랜드의 관계를 표시하였다.

2.2.2.1 오퍼레이터(operator)

CD, DFD에서는 버블로 표시되는 개개의 처리로부터 오퍼레이터를 생성할 수 있다. 또한 자료 저장소로 향하는 자료 흐름 및 자료 저장소로부터 나오는 자료 흐름은 여러가지 화일의 쓰기(write) 및 화일로부터의 읽기(read)를 처리하는 것이므로 오퍼레이터라고 할 수 있다. MS에서는 각 S_i에 포함된 동사인 v가 하나의 처리를 나타내고 있으므로 오퍼레이터로 볼 수 있으며, DD에서는 데이타 구조를 정의하는데 사용되고 있는 연산자를 오퍼레이터로 볼 수 있다.

2.2.2.2 오퍼랜드(operand)

CD, DFD에서는 자료 흐름에 나타나는 데이타를 오퍼랜드에 대응시킬 수 있으며, 자료 저장소는 일시적인 화일이나 데이타베이스 또는 다른 어떤 자료의 출력 형태로 생각할 수 있으므로 각 자료 저장소에 포함된 데이타를 오퍼랜드라고 볼 수 있다. MS에서 S_i에 포함된 d_a, d_b 및 S_i에 포함된 d_i를 구성하고 있는 각 최소 단위 데이타를 오퍼랜드라고 할 수 있으며, DD에서는 기술되어 있는 각각의 데이타 전체를 오퍼랜드로 볼 수 있다.

〈표 2〉 구조화 명세서의 오퍼레이터와 오퍼랜드
〈Table 2〉 Operator and Operand of Structured Specification

명세서	오퍼레이터	오퍼랜드
DFD	버블로 표시된 처리 자료 저장소의 읽기/쓰기	자료 흐름상의 데이타 자료 저장소의 구성 원소
MS	문장에 포함된 동사(v)	d_a, d_b, d_c의 최소 단위 데이타
DD	자료 사전에 사용되는 연산자	기술되어 있는 모든 데이타

3. 구조화 명세의 평가척도

3.1 구조화 명세의 정의

구조화 명세 H에 포함된 기본 기능에 대응하는 전체 처리를 통해서 구성된 DFD를 $D_H(H)$, 전체 MS의 기술된 내용을 합친 것을 $Ms_H(H)$, H에 포함된 전체 DD의 기술내용을 중복이 없도록 합친 것을 $Dd_H(H)$라고 표현한다.

3.1.1 DFD

임의의 DFD를 D라고 한다면 다음과 같이 정의한다.

〈표 3〉 DFD의 정의
〈Table 3〉 Definition of DFD

기 호	의 미
$b(D)$	D에 포함된 처리 수
$v(D)$	D에 포함된 각 처리에 대한 입력차수의 총합
$n(D)$	D에 포함된 데이타 항목 종류 수
$f(D)$	D에 포함된 자료 흐름 수
$s(D)$	D에 포함된 자료 저장소 수
$a(D)$	D에 포함된 자료 저장소 차수의 총합
$a_w(D)$	D에 포함된 자료 저장소 입력차수의 총합
$a_r(D)$	D에 포함된 자료 저장소 출력차수의 총합
$I(D)$	D에 포함된 입력차수 0의 연결점 수
$O(D)$	D에 포함된 출력차수 0의 연결점 수

3.1.2 DD

임의의 DD를 D_d라고 한다면 다음과 같이 정의한다.

〈표 4〉 DD의 정의
〈Table 4〉 Definition of DD

기 호	의 미
$n(Dd)$	Dd의 데이타 종류 수
$f(Dd)$	Dd의 데이타 출현회수

3.1.3 MS

임의의 MS를 M_s라고 한다면 다음과 같이 정의한다.

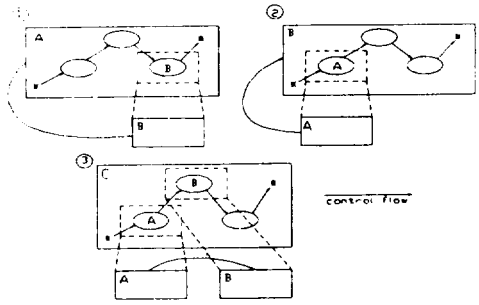
〈표 5〉 MS의 정의
〈Table 5〉 Definition of MS

기 호	의 미
$Sn(Ms)$	자연어로 기술된 Ms에 포함된 문장의 수
$S(Ms)$	$M(Ms)$의 문(satement)의 수
$b(Ms)$	$M(Ms)$에 포함된 v의 종류 수
$v(Ms)$	$M(Ms)$의 v의 출현회수
$n(Ms)$	$M(Ms)$에 포함된 최소 단위 데이타의 종류 수
$f(Ms)$	$M(Ms)$의 최소 단위 데이타의 출현회수
$s(Ms)$	$M(Ms)$가 참조/변신하는 자료 저장소 수
$a(Ms)$	$M(Ms)$의 자료 저장소 참조/변신 회수의 총합
$a_w(Ms)$	$M(Ms)$의 자료 저장소 변신 회수의 총합
$a_r(Ms)$	$M(Ms)$의 자료 저장소 참조 회수의 총합
$I(Ms)$	$M(Ms)$에 입력되는 최소 단위 데이타 수
$O(Ms)$	$M(Ms)$에서 출력되는 최소 단위 데이타 수

3.2 컴포넌트(component)

구조화 명세를 구성하는 DFD 및 MS를 구조화 명세의 컴포넌트라고 부르며, H에 포함된 컴포넌트 전체의 집합을 $\Gamma(H)$라 한다. 컴포넌트 A와 B 사이에 다음과 같은 관계가 있을 때 A로부터 B로의 제어흐름(control flow)이 존재한다고 볼 수 있으며, (그림 3)은 제어 흐름의 관계를 나타낸 것으로 각각에 대한 설명은 다음과 같다.

- A가 처리 B의 부모이다(그림 3의 ①).
- B가 처리 A의 부모이고, B에 포함된 처리 A의 출력 자료 흐름이 B의 다른 처리의 입력 자료 흐름으로 존재한다(그림 3의 ②).
- 처리 A, 처리 B의 공통 부모가 C이고, C에 포함된 처리 A의 출력 자료 흐름이 처리 B의 입력 자료 흐름으로 존재한다(그림 3의 ③).



(그림 3) 컴포넌트간의 제어흐름 조건
(Fig. 3) Control Flow Condition of Component

3.3 구조화 명세의 메트릭스

프로그램 코드의 품질과 규모를 정량적으로 평가하기 위한 척도는 많이 제안되어 있다. 그러나 구조화 명세에 관한 평가척도는 제안되어 있는 것이 없으므로 구조화 명세에 대한 척도를 프로그램 코드의 척도에 적용하여 품질과 규모의 정량적인 평가를 처리할 수 있는 척도를 제안한다.

3.3.1 시스템 규모 및 모듈에 관한 메트릭스

모듈의 개수와 코드 메트릭스의 모듈 사이즈 등에 관한 메트릭스를 대응시킨다.

- 컴포넌트 수 $M(H)$: 코드 메트릭스의 모듈 수

$$M(H) = |\Gamma(H)| \qquad (3)$$

- 시스템 규모 $S(H)$: 코드 메트릭스의 라인 수

$$S(H) = \sum_{C \in \Gamma(H)} v(C) \qquad (4)$$

· 컴포넌트의 규모 평균 $Sm(H)$: 코드 메트릭스의 모듈 사이즈

$$Sm(H) = S(H)/M(H) \quad (5)$$

3.3.2 정보 흐름 메트릭스(Information Flow Metrics)

정보 흐름 메트릭스[7]는 시스템 요소에서의 정보 흐름에 기반을 둔 것으로, 임의의 컴포넌트 C에 관한 정보 흐름 메트릭스 $If(C)$는 (식 6)과 같이 정의한다.

$$If(C)=v(C)\times(Fi(C)\times Fo(C))^2,\ C\in\Pi(H) \quad (6)$$

단, $Fi(C)$: C에 들어온 제어흐름 수와 C가 참조하고 있는 자료 저장소 수의 합(fan-in)

$Fo(C)$: C로부터 나온 제어흐름 수와 C가 변신시키는 자료 저장소 수의 합(fan-out)

3.3.3 Halstead의 소프트웨어 과학에 관한 메트릭스

구조화 명세에서 추출할 수 있는 오퍼레이터와 오퍼랜드를 Halstead의 메트릭스[8]에 적용한다면 분석단계에서 개발해야 할 소프트웨어의 규모와 노력정도를 알 수 있다.

〈표 6〉 소프트웨어 과학의 메트릭스
〈Table 6〉 Metrics of Software Science

내 용	메 트 릭 스
H의 오퍼레이터 종류 수	$n_1(H) = b(D_1) + b(Ms_n)$
H의 오퍼랜드 종류 수	$n_2(H) = n(Dd_1)$
H의 오퍼레이터 출현회수	$N_1(H) = v(D_p) + v(Ms_n)$
H의 오퍼랜드 출현회수	$N_2(H) = f(D_p) + f(Ms_n) + f(Dd_1)$
H의 어휘	$n(H) = n_1(H) + n_2(H)$
H의 길이	$N(H) = N_1(H) + N_2(H)$
H의 길이 견적	$L(H) = n_1(H)\log_2 n_1(H) + n_2(H)\log_2 n_2(H)$
H의 볼륨	$V(H) = N(H)\log_2 n(H)$
H의 난이도	$D(H) = [n_1(H)\cdot N_2(H)]/(2\cdot n_2(H))$
H의 수준	$L(H) = 1/D(H)$
H의 노력도	$E(H) = V(H)/L(H)$

3.3.4 McCabe의 사이크로매틱(cyclomatic) 수에 관한 메트릭스

프로그램의 논리적인 복잡도를 나타내는 메트릭스로 사이크로매틱 수를 구하는 방법[9]중에서 predicate 수에 의해 구하는 방법을 고려하였다. 그리고 이것을 구조화 명세에 적용하기 위해서 각 모듈의 처리 절차를 나타내고 있는 MS를 이용하여 제어구조의 복잡도를 나타내었다. 단일 조건문과는 달리 다중 조건문인 경우는 조건문의 수만큼 경로가 생기므로 해당 조건문 수가 자체의 복잡도를 나타낸다. 따라서 MS에서 사이크로매틱 수를 관한 메트릭스 $\rho(\mathrm{Ms})$는 단일 조건문의 수와 다중 조건문의 수를 모두 고려하였으며, (식 7)과 같이 정의한다.

$$\rho(Ms)=M(Ms)-d_{c1}+M(Ms)-d_{c2}+1 \quad (7)$$

단, $M(Ms)-d_{c1}$: if, while, until의 단일 조건문의 수

$M(Ms)-d_{c2}$: when, case에서 otherwise, default를 제외시킨 다중 조건문의 수

4. 공수 예측을 위한 실험적 평가

4.1 의사코드 변환 방법

본 논문에서는 공수 예측을 위한 실험적 평가를 하기 위해 자료 흐름도를 의사코드로 변환하는데 걸리는 시간을 계측한 결과를 분석하는 방법을 사용하였다. 우선 자료 흐름도를 의사코드로 변환하는 절차를 다음과 같이 정하였다.

단계 1: DFD n의 절차명은 Fn으로 한다.

단계 2: DFD n으로 입력되는 자료흐름과 출력되는 자료흐름은 Fn의 입출력 인수 리스트에 기술하며, 단 출력 인수 앞부분에는 '*'를 표시한다.

단계 3: 자료 저장소가 존재하는 경우에는 Fn의 문장 첫머리에 fopen( )문을 사용해서 화일을 open 시킨다.

단계 4: 화일로부터 값을 읽거나 화일에 값을 쓸 때는 read( )문 및 write( )문을 이용한다. 예를 들어 자료 저장소 X로부터 x를 읽거나 자료 저장소 Y에 y를

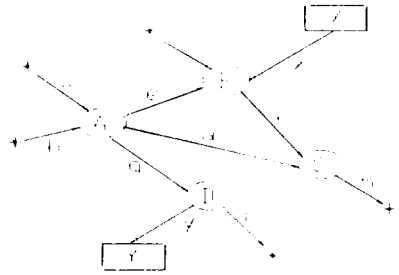
쓸 경우 read(X, x)와 write(Y, y)로 기술한다. 단, read는 처리에서 데이타를 사용하기 전에 write는 처리에서 데이타가 체크된 후에 기술한다.

단계 5: 각 처리에 대응한 절차는 그 처리의 이름을 절차명으로 하고, 그 처리에 들어오는 자료 흐름을 입력, 처리로부터 나가는 자료 흐름을 출력으로 한다. 입력 및 출력은 인수 리스트로 나란히 기술하며, 출력은 앞에 '&'를 사용하여 기술한다. 단, Fn의 전체 출력에 대응하는 데이타는 '&'를 사용하지 않는다.

단계 6: 각 절차는 입력 데이타의 흐름 순서에 의해 실행되어야 한다.

단계 7: 전체 화일로의 액세스에 관한 기술을 종료한 후 fclose( )문을 이용해서 화일을 close한다.

이러한 의사 코드 변환 절차에 적용하여 (그림 4)의 DFD는 (그림 5)와 같은 결과를 얻을 수 있다.



(그림 4) DFD 1의 예제
(Fig. 4) Example of DFD 1

```
1 :  F1(a, b, c, *f, *m)                    //단계 1, 2
2 :  {
3 :      fopen(X) ;                         //단계 3
4 :      fopen(Y) ;                         //단계 3
5 :      A(a, b, &d, &e, &g) ;              //단계 5, 6
6 :      read(X, x) ;                       //단계 4
7 :      B(c, e, x, &f) ;                   //단계 5, 6
8 :      C(d, f, m) ;                       //단계 5, 6
9 :      D(g, &y, f) ;                      //단계 5, 6
10 :     write(Y, y) ;                      //단계 4
11 :     fclose(X) ;                        //단계 7
12 :     fclose(Y) ;                        //단계 7
13 : }
```

(그림 5) DFD 예제의 의사코드 결과
(Fig. 5) Pseudo Code Result of DFD Example

4.2 실험 1

실험 데이타를 분석하기 위해 통계학적인 방법인 분산분석(ANOVA : Analysis of Variance) 방법을 사용하였고, 결과를 얻기 위해 통계 패키지인 SAS를 이용하였다.

4.2.1 실험조건 및 방법

동일한 구조를 갖는 DFD에서도 표현방법에 의해서 코딩의 공수에 차이가 발생하는지를 알아보기 위해 자료 흐름 방향이 일정한 상태와 자료 흐름의 교점 유무, 자료 저장소의 유무를 요인으로 채택하여 실험하였다. 실험 대상자는 전자계산학을 전공하고 있는 대학원생 및 학생 10명을 대상으로 하였다. 실험 대상자는 의사코드의 변환 절차를 통해 DFD를 코드로 변환하는데 필요한 시간을 계측하였으며, 각 DFD의 요인별 내역은 〈표 7〉과 같으며, 10명에 관한 코드 변환시 걸리는 계측시간의 결과를 〈표 8〉에 나타내었다.

〈표 7〉 실험 1 : DFD의 조건 내역
(Table 7) Experiment 1 : Condition Contents of DFD

DFD 명	자료 흐름의 방향	자료 흐름의 교점	자료 저장소
DFD1-1	일정방향	없 음	없 음
DFD1-2	일정방향	있 음	있 음
DFD1-3	다 양	없 음	있 음
DFD1-4	다 양	있 음	없 음

〈표 8〉 실험 1 : 코딩 결과
(Table 8) Experiment 1 : Coding Result

실험대상자	코딩시간(초)			
	DFD1-1	DFD1-2	DFD1-3	DFD1-4
P1	72	171	215	116
P2	116	151	253	118
P3	192	211	226	168
P4	185	175	228	183
P5	126	163	242	321
P6	71	123	221	195
P7	90	150	175	105
P8	90	180	190	105
P9	97	228	182	204
P10	150	286	215	139

4.2.2 분산 분석 결과

10인×4=40개의 데이타를 이용해서 자료 흐름의 일정 방향 정도, 자료 흐름의 교점 유무, 자

료 저장소의 유무에 관한 인자의 영향도를 나타내기 위한 분산 분석 결과는 〈표 9〉와 같다.

분산분석은 이름이 뜻하는 것 처럼 특정 값의 분산 또는 산포를 분석하는 방법으로 특정 값의 산포를 제곱합(sum of square)으로 나타내고, 이 제곱합을 실험에 관련된 요인별로 분해하여, 순수한 오차(error)에 의한 영향보다 큰 영향을 주는 요인이 어떤 것인가를 찾는 분석 방법이다. 그리고 분산분석은 귀무가설을 인자에 대한 처리 효과가 모두 같다고 가정하고 이 귀무가설을 기각할 경우 각 인자에 대한 유의차가 존재한다고 하여 그 인자에 의한 처리 효과가 있음을 검정하는 방법이다. 즉, F_0 비의 검정 통계량이 F 분포표 값보다 클 경우에 귀무가설을 기각시킴으로 각 요인에 대한 처리 효과를 검정한 것으로 다음과 같은 검정 통계량과 기각역에 의해 결과를 얻을 수 있다.

- 검정통계량 : $F_0 = \frac{\text{각 요인의 제곱평균}}{\text{잔차의 제곱평균}}$
- 기 각 역 : $F_0 \geqq F$(각 요인의 자유도, 잔차의 자유도 ; 유의수준)

분산분석 결과를 통해 자료 흐름의 교점 유무에 관해서는 유의차가 없었으며, 자료 흐름의 일정방향 정도와 자료 저장소의 유무에 관해서는 유의차가 있었다. 따라서 이 결과 자료 흐름의 일정 방향과 자료 저장소의 존재 유무가 코딩의 공수에 영향을 주고 있음을 알 수 있었으며, 자료 저장소에 관한 보다 자세한 영향도는 실험 2에서 고찰한다.

〈표 9〉 실험 1 : 분산 분석 결과
〈Table 9〉 Experiment 1 : ANOVA Result

요 인	제곱합	자유도	제곱평균	F_0 비	비교	F 분포값
자료흐름의 일정 방향	14976.9	1	14976.90	7.75	>	F(1,27:0.01)=7.68 F(1,27:0.05)=4.21
자료흐름의 교점 유무	608.4	1	608.40	0.31	<	F(1,27:0.01)=7.68 F(1,27:0.05)=4.21
자료저장소의 유 무	32604.1	1	32604.10	16.87	>	F(1,27:0.01)=7.68 F(1,27:0.05)=4.21
잔 차	52195.6	27	1933.17			
총 변 동	130700.4	39				

4.3 실험 2

4.3.1 실험조건 및 방법

구조화 명세와 프로그램 코드의 대응에 의하면 연결점 수는 함수의 인수와 리턴 값의 인수에 대응하고, 자료 저장소는 화일 액세스의 절차에 대응하며, 처리 수는 함수의 기능을 나타낼 수 있다. 따라서 DFD를 기초로 코딩을 처리할 때의 공수와 생성된 모듈의 사이즈 등을 예측하기 위해서 처리 수와 실험 1의 결과에 의한 자료 저장소 수 및 노드에 대응하는 연결점 수가 코딩의 공수에 영향을 주는지를 고려하는 것이 필요하다. 따라서 본 실험에서는 처리 수, 연결점 수 및 자료 저장소 수가 코딩의 공수에 영향을 주는지의 여부를 조사한다. 실험 1과 동일한 방법으로 실시하며 각 DFD의 내역은 〈표 10〉과 같으며, 코드 변환시 걸리는 계측 시간의 결과는 〈표 11〉과 같다.

〈표 10〉 실험 2 : DFD의 조건 내역
〈Table 10〉 Experiment 2 : Condition Contents of DFD

DFD 명	처 리 수	연 결 점 수	자료 저장소 수
DFD2-1	3	3	0
DFD2-2	3	5	1
DFD2-3	3	7	2
DFD2-4	5	3	1
DFD2-5	5	5	2
DFD2-6	5	7	3
DFD2-7	7	3	2
DFD2-8	7	5	0
DFD2-9	7	7	1

〈표 11〉 실험 2 : 코딩 결과
〈Table 11〉 Experiment 2 : Coding Result

실험대상자	코딩시간(초)				
	DFD2-1	DFD2-2	DFD2-3	DFD2-4	DFD2-5
	DFD2-6	DFD2-7	DFD2-8	DFD2-9	
P1	24	59	100	124	118
	59	150	60	98	
P2	20	75	97	98	160
	64	142	54	121	
P3	25	47	184	101	140
	39	203	50	121	
P4	31	102	134	154	131
	77	158	78	153	
P5	34	65	118	91	131
	53	253	77	139	
P6	45	73	178	117	159
	75	192	65	122	
P7	20	80	160	150	150
	65	170	70	130	
P8	16	60	100	85	180
	60	130	50	105	
P9	32	80	174	80	139
	68	155	107	122	
P10	23	51	96	82	92
	50	115	50	88	

4.3.2 분산 분석 결과

10인×9=90개의 데이타를 이용해서 분산 분석을 처리하며 처리 수, 연결점 수, 자료 저장소 수에 관한 인자의 영향도를 나타내기 위한 분산 분석 결과는 〈표 12〉와 같다. 분산 분석을 통해 처리 수와 자료 저장소 수에 관해서는 유의차가 있었으나, 연결점 수에 관해서는 유의차가 없었다.

〈표 12〉 실험 2 : 분산 분석 결과
〈Table 12〉 Experiment 2 : ANOVA Result

요 인	제곱합	자유도	제곱평균	F_0 비	비교	F분포값
처 리 수	25702.69	2	12851.34	27.57	>	F(2,74;0.01)≒4.92 F(2,74;0.05)≒3.13
연결점 수	2731.09	2	1365.54	2.93	<	F(2,74;0.01)≒4.92 F(2,74;0.05)≒3.13
자료 저장소 수	137090.49	2	68545.24	147.04	>	F(2,74;0.01)≒4.92 F(2,74;0.05)≒3.13
잔 차	34496.18	74	466.16	-		
총 변 동	215435.12	89	-	-		

$F(2,70;0.05)=3.13$, $F(2,70;0.01)=4.92$, $F(9,70;0.05)=2.02$, $F(9,70;0.01)=2.67$

따라서 실험 결과를 통해 코딩의 공수 예측에 있어서는 처리 수와 자료 저장소 수를 고려해야 함을 알 수 있었다. 처리 및 자료 저장소의 존재와 그 개수가 코딩의 공수에 영향을 주는 것이 명확하지만 여러가지 측면에서 비교하여 영향을 주는지의 여부를 실험 데이타를 기초로 하여 조사하는 것이 필요하며, 본 실험의 경우 자료 저장소 1개(입력차수, 출력차수 공히 1개)는 처리 4개에 해당함을 인식해야 한다. 본 실험에 있어서 처리 수 1개는 코드 중에 있는 처리 호출이 1행에 대응하고 있으나, 자료 저장소 1개는 화일의 open, 화일로부터 데이타 읽기, 화일로의 데이타 쓰기, 화일의 close의 4행에 대응하고 있다. 이를 바탕으로 임의의 DFD인 D에 대응하는 코딩의 공수 예측 모델 $Effort(D)$는 (식 8)과 같이 나타낼 수 있다.

$$Effort(D)=\alpha b(D)+\beta s(D)+\gamma a(D) \qquad (8)$$

단, α: 처리 1개에 해당하는 코딩 공수
β: 1개의 자료 저장소에 대응하는 1개의 화일의 open/close 코딩에 필요한 공수
γ: 1개의 화일 액세스(read/write)의 코딩에 필요한 공수

여기에서는 2종류의 화일 액세스의 처리를 동일하게 했으나 이 점을 좀더 고려해야 할 필요가 있다. DFD에 있어서 자료 저장소는 데이타의 일시적인 보관 장소 또는 자료 흐름 순서를 변경하는 장소이다. DFD 상에서 표기된 자료가 동일하더라도 같은 값을 갖는 자료라고 볼 수 없으므로 쓰기(write)와 읽기(read)의 각각의 코딩 공수는 다르게 고려되어야 한다. 따라서 (식 8)의 공수 예측 모델 $Effort(D)$는 (식 9)와 같이 변경될 수 있다.

$$Effort(D)=\alpha b(D)+\beta s(D)+\gamma_r a_r(D)+\gamma_w a_w(D) \qquad (9)$$

단, $a_r(D)$: D에 의한 자료 저장소의 출력차수의 총합
$a_w(D)$: D에 의한 자료 저장소의 입력차수의 총합
$\gamma_r(D)$: 자료 저장소로부터 읽는데 필요한 코딩 공수
$\gamma_w(D)$: 자료 저장소로 쓰는데 필요한 코딩 공수

또한 본 실험에서 사용한 DFD로부터 프로그램 코딩으로의 변환 규칙을 데이타의 구조와 제어구조 등을 고려한 상에서 엄밀히 정의할 수 있다면, 완성된 프로덕트 사이즈의 예측을 처리하는 것도 가능하다고 볼 수 있다. 임의의 DFD인 D를 기초로 생성된 모듈의 사이즈(LOC, 바이트 수 등의 가산적인 것)의 예측 모델을 $Volume(D)$로 하면 (식 10)과 같이 정의할 수 있다.

$$Volume(D)=Ab(D)+Bs(D)+\Gamma_r a_r(D)+\Gamma_w a_w(D) \qquad (10)$$

단, A : 처리 1개에 요구되는 코드 사이즈
B : 화일의 open/close에 요구되는 코드 사이즈
Γ_r: 화일로부터 데이타를 읽는데 요구되는 코드 사이즈
Γ_w: 화일로 데이타를 쓰는데 요구되는 코드 사이즈

이상으로 DFD에 의한 하위 공정에 관한 공수와 프로덕트의 사이즈를 예측하기 위해서 자료 저장소와 처리 수를 이용하여 공수 예측 모델을 제안하였다.

6. 결 론

본 연구에서는 개발초기 단계인 상위공정 단계에서 하위공정 단계의 품질 규모를 예측하기 위해서 상위공정의 결과물을 분석하고 규모에 영향을 줄 수 있는 정량적인 요인을 분석하였다. 이를 위해 구조화 분석법에서 생성되는 구조화 명세인 자료 흐름도, 자료 사전, 소단위 명세서를 이용하였다.

첫째, 구조화 명세를 구성하는 3종류의 문서에 관련된 동일 특성의 정량적 평가 요소를 추출하여 정의하였다.

둘째, 구조화 명세에 관한 오퍼레이터와 오퍼랜드를 정의하였다.

셋째, 프로그램의 코드 메트릭스에 구조화 명세의 정량적인 평가 요소를 대응시킴으로써 품질과 규모를 정량적으로 평가할 수 있는 메트릭스를 제안하였다.

넷째, 자료 흐름도를 코드로 변환하는데 걸리는 시간의 요인별 분석을 통해 하위공정에 관한 DFD의 공수 예측 모델을 유도하였다.

향후 연구과제로는 의사코드 변환에서는 없었지만 실제의 프로젝트를 대상으로 하여 데이타를 분석하고 실제로 적용가능하도록 상세한 예측 모델을 구축하는 것과 이러한 절차를 통해 유도한 예측 모델에 관한 각각의 계수를 결정하고, 자료 사전과 소단위 명세서에 관한 예측 모델도 실험을 통해 명확히 하는 것이 필요하다.

참 고 문 헌

[1] Boehm, B. W., "Software Engineering Economics", Prentice Hall, Inc. 1981.

[2] DeMarco, T., "Structured Analysis and System Specification", Yourdon Press, 1978.

[3] DeMarco, T., "Controlling Software Projects: Management, Measurement & Estimation", Yourdon Press, 1982.

[4] Yourdon, E., "Modern Structured Anaysis", Prentice Hall, 1989.

[5] Martin, J., and McClure, C., "Structured Techniques for Computing", Prentice-Hall Inc., 1985.

[6] Pressman, R., "Software Engineering: A Practitioner's Approach", McGraw-Hill, 1982.

[7] Henry, S. and Kafura, D., "Software Structure Metrics Based on Information Flow", IEEE Trans. SE., Vol. 7, No. 5, Sept. 1981.

[8] Halstead, M., "Element of Software Science", Elsevier, North-Holland, 1977.

[9] McCabe, T. J., "A Complexity Measure", IEEE Trans. SE., Vol. 2, No. 4, pp. 308-320, 1976.

[10] Babll, R. G., "Data-Driven Implemantation of Data Flow Diagrams", proc. 6th ICSE, pp. 309-318, Sept. 1982.

[11] Pong, L. and Tse, T. H., "Towards Formal Foundation for DeMarco Data Flow Diagrams", The Computer Journal, Vol. 32 (1), pp. 1-12, 1989.

[12] Putnam, H. H., "A General Empirical Solution to the Macro Software Sizing and Estimating Problems", IEEE Trans. SE., Vol. 4, pp. 345-361, 1978.

[13] Ross, D. T., "Structureed Analysis(SA): A Language for Communication Ideas", IEEE Trans. SE., Jan. 1977.

[14] Albrecht, A. and Gaffeny, J., "Software Function Source Lines of Code and Development Effort Prediction: A Software Science Validation", IEEE Trans, SE. Vol. 9,

No. 6, 1983.

[15] Woodfield, S. N., Shen, V. Y. and Dunsmore, H. E. "A Study of Several Metrics for Programming Effort", IEEE Trans. SE Vol. 2, 1981.

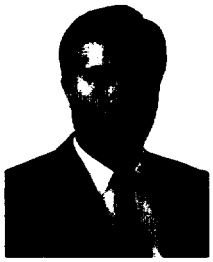
[16] 大筆豊, "ソフトウェアの コスト見積り技術", IPS, Vol. 33. No. 8, 1992.

[17] 坪井信男 외 6인, "工數豫測モデルに基づく定量的工程管理の試み", ソフトウェア工學, 1992.

[18] 立田種宏, "リアルタイムSA手法による要求仕様の設計仕様への變換", ソフトウェア工學, 1988.

김 명 옥

1991년 강원대학교 자연과학대학 전자계산학과 졸업(이학사)
1993년~94년 강원대학교 전자계산학과 소프트웨어공학 전공(이학석사)
1991년~92년 아시아 엔지니어링(주) 근무
1995년~현재 한컴데이타(주) 기술연구소 연구원
관심분야: 소프트웨어 공학(소프트웨어 품질평가, 객체지향 분석과 설계, 객체지향 프로그래밍).



백 청 호

1968년 서울대학교 사범대학 수학교육과 졸업(학사)
1981년 성균관대학교 경영대학원 정보처리 전공(경영학석사)
1992년 경기대학교 대학원 경영정보학 전공(경영학박사)
1993년~94년 캐나다 토론토대학교 객원교수
1994년~현재 한국정보처리학회 감사
1984년~현재 강원대학교 전자계산학과 부교수
관심분야: 수치해석, 알고리즘 해석, 계산이론 및 그래프 이론, 경영정보시스템.

양 해 술

1975년 홍익대학교 공과대학 전기공학과 졸업(학사)
1978년 성균관대학교 정보처리학과 정보처리 전공(석사)
1991년 日本 오사카대학 기초공학부 정보공학과 소프트웨어공학 전공(공학박사)
1975년~79년 육군중앙경리단 전자계산실 시스템분석장교
1984년~95년 성균관대학교 경영대학원 강사
1986년~87년 日本 오사카대학 객원연구원
1993년~94년 한국정보과학회 학회지 편집부위원장
1994년~현재 한국산업표준원(IIS) 이사
1994년~현재 한국정보처리학회 논문편집위원장
1980년~현재 강원대학교 전자계산학과 교수
관심분야: 소프트웨어 공학(특히, S/W 품질보증과 평가, SA/SD, OOA/OOD/OOP, CASE, SI), 소프트웨어 프로젝트관리.