소프트웨어 사업대가기준 보정계수의 유도 및 민감도 분석

변 분 희[†] · 권 기 태[††]

요 약

성공적인 소프트웨어 프로젝트 수행을 결정짓는 중요한 작업 중의 하나는 프로젝트 개발 초기에 소프트웨어 개발 비용을 정확하게 산정하는 것이다. 정확한 산정을 위해서는 빠르게 변화하고 있는 개발 환경 및 기술 변화에 따른 변경 요인을 비용 산정 과정 시 반영시켜야 하며 이를 위해서는 적절한 보정계수 선정과 보정계수 값 적용이 중요시된다. 이에 본 논문에서는 소프트웨어 개발비 대가기준을 위한 어플리케이션 유형 보정계수와 언어 보정계수 산정을 위해 AHP 기법을 적용하였다. 또한, 민감도 분석을 통하여 각 판단 기준이 보정계수에 미치는 영향을 조사하였다.

결론적으로, 어플리케이션 유형 보정계수 값은 처리 복잡도에 비해 데이터관리 복잡도와 제어 복잡도에 민감하게 반응하며 또한, 언어 보정계수 값은 디버깅 시간에 비해 개발인력의 보편성과 개발코딩 시간에 더 민감하게 반응하였다.

향후, 연구과제로는 국내의 소프트웨어 개발 환경과 기술을 적절히 반영시킬 수 있는 추가적인 보정계수 결정과 적절한 보정계수 값 조정에 대하여 연구할 것이다. 또한, 언어 보정계수에서는 구체적인 프로그래밍 언어를 들어 좀더 세부적으로 구분한 후 보정계수를 산정하고자 한다.

키워드 : 소프트웨어 비용 산정, 소프트웨어 사업대가기준, AHP, 보정 계수, 민감도 분석

A Study on the Derivation and Sensitivity Analysis of the Adjustment Factor in the Software Cost Estimation Guidelines

Boon Hee Byun[†] · Ki Tae Kwon[††]

ABSTRACT

One of the most significant tasks of software development project is to know how much it will be the software development cost in the early stage of software development cycle. The software development environment and technology are changing very rapidly. For accuracy, we should apply those to the software cost estimation. And it is important that we select the suitable adjustment factor and the value of a suitable adjustment factor. For that, this paper have applied the method of AHP. And we have also analyzed the sensitivity of the adjustment factor which is influenced by decision metrics.

In conclusion, the value of the application type adjustment factor is responded more sensitively to the data complexity and the control complexity than processing complexity. And the value of the language adjustment factor is responded more sensitively to the supplying manpower and the time of the coding than the time of the debugging.

In the future, we will research the selection of an additional adjustment factor and a suitable value of the adjustment factor which are influenced by the environment and the technology of the domestic software development. And then, in the language adjustment factor, we will try to calculate the value about the individual programming language.

Key Words : Software Cost Estimation, Software Cost Estimation Guidelines, Analytic Hierachy Process, Adjustment Factor, Sensitivity Analysis

1. 서 론

성공적인 소프트웨어 프로젝트 수행을 결정짓는 중요한 작업 중의 하나는 소프트웨어 개발비용을 정확하게 산정하는 것이다.

성공적인 소프트웨어 프로젝트가 되느냐 실패 또는 재앙이 된 프로젝트가 되느냐를 결정하는 중요한 차이 중에 하나는 프로젝트 시작 초기에 비용, 자원, 품질 등을 평가하는 방법에 달려 있다. 스케줄 또는 비용 산정이 과도하게 계획된 프로젝트는 스케줄 또는 비용 산정 시 적절한 방법을 사용하지 않으므로 인해 종종 발생된다. 실패한 프로젝트의 근본적인 문제는 부정확한 계획과 산정에서 비롯된다[1].

소프트웨어 비용 산정 모델에 관한 주요 연구는 1965년 169개 소프트웨어 프로젝트의 104가지의 속성에 관한 SDC

※본 연구의 일부는 정보통신부의 정보통신연구개발사업(한국전산원 위탁과제: NCA VI-PER-03055)의 연구결과임.
† 준 회 원 : 강릉대학교 컴퓨터공학과 박사수료
†† 종신회원 : 강릉대학교 컴퓨터공학과 교수
논문접수 : 2007년 3월 14일, 심사완료 : 2007년 10월 13일

DOI: 10.3745/KIPSTD.2008.15-D.1.61

의 광범위한 연구로 시작되었다[2].

1970년대는 직접적인 산정 값을 얻기 위해 미리 정의 된 여러 가지 비용인자들을 포함하는 방식으로 대표적인 모델로 Wolverton, Walston-Felix, Putnam, Albrecht등을 들 수 있다. 이 시기에는, 실제적인 경험을 통해 프로젝트 크기, 비용인자 또는 전문가의 판단이 중요하다는 것을 알게 되었다. 특히, Conte는 분석적인 산정식, 통계적인 데이터의 적합성, 전문가의 판단 등을 결합하는 방법이 요구됨을 강조하였고, 이러한 방법들은 전문가에 의해 만들어진 명목적인 산정을 위한 조정을 고려하였다. 대표적인 유형으로 1981년 Boehm에 의해 제안된 COCOMO(Constructive Cost Model)모델이 있다. 기본적인 노력 인자로써 시스템 크기를 이용하는 등식을 제공함과 함께 예측된 개발 노력은 15가지 비용인자들을 추가하여 조정한다. 이러한 방법의 다른 예로 SLIM(Putnam, 1978)과 COPMO(Conte etal, 1986)등이 있다.

1980년대에는 매개변수 방법이 폭넓게 이용되었으며, 이 시기의 모델들은 다양한 규모와 환경적인 데이터 집합을 이용하여 비교하였다. 이러한 연구에서 얻은 주요한 결론은 비용 산정 모델들은 환경이 다른 경우, 측정하지 못한 인자들이 적용된다면 성능이 떨어진다는 것이다. 대표적인 연구자로 Kitchenham and Taylor(1985), Conte et al(1986), Kemerer (1987)등을 들 수 있다.

1990년에는 Abdel-Hamid와 Madnick 같은 연구자들은 소프트웨어 개발은 복잡한 동적인 프로세스이며 복잡함과 생산성에서 나타나는 다양성을 설명할 수 있는 변경과 관련된 관계성을 거의 알지 못함을 알게 되었다. 따라서, 1990년대에는 Optimized Set Reduction(Briand et al., 1992), 인공 신경망 (Jorgensen, 1995); Finnie at al., 1997), CART regression trees (Srinivasan and Fisher, 1995; Kitchenham, 1998), 그리고 Analogy-based estimation(Mukhopadyay et al., 1992; Sheppered and Schofield, 1997; Walkerden and Jeffery, 1999)등과 같은 기계 학습 알고리즘에 기반한 비모수 모델링 기법의 소개 및 산정법등이 등장하였다.

또한, 전문가의 판단과 히스토리컬 데이터를 가지고 전문가의 의견을 결합시키는 방법이 연구되고 비교되어지고 있다. 예를 들어, 주관적인 노력 산정(Hőst and Wohlin, 1998; Stensrud and Myrtveit, 1998) 전문가의 지식에 기반한 모델링, 전문가의 의견과 프로젝트 데이터를 결합한 기술들을 들 수 있다. 이러한 접근법은 앞으로 소프트웨어 비용 산정 모델의 중요한 요인이 될 것이다[3].

국내의 소프트웨어 비용 산정 모델에 관한 연구로는 소프트웨어 개발비 대가기준을 들 수 있다. 소프트웨어 개발비 대가기준은 1989년에 최초로 고시된 후, 소프트웨어 산업 발전에 맞춰 꾸준히 개선 연구를 추진하여 현재 정부 및 공공, 민간에서 널리 활용되고 있는 실정이다. 그러나 소프트웨어 개발비 대가기준은 과거 10여년 이상 본수 및 스텝 수 기반의 규모측정 방식이 사용되어 객관적이고 신뢰성 있는 비용 산정이 불가능하였다. 또한 객관적인 근거 없이 설정된 보정계수 및 비용 산정 구조의 영향으로 산정 결과에 대한 논란이 지속적으로 제기되었다.

이에 합리적인 소프트웨어 개발비 산정을 위해 2003년 정보통신부, 한국전산원, 한국 소프트웨어 산업협회를 중심으로 사용자 요구를 반영하는 기능점수 기반의 소프트웨어사업 대가기준 개선연구가 진행되었고, 그 결과 2004년 기능점수 방식에 따라 소프트웨어 개발비 대가기준이 개정 고시되었다[4].

2004년 고시된 개정 소프트웨어 개발비 대가기준은 선진 모형과 비교하여 볼 때, 일부 기존 보정계수의 조정 및 신규 보정계수의 도입을 중심으로 개선 필요성이 제기되고 있다[5].

이에 본 논문에서는 AHP 기법을 이용하는 어플리케이션 유형 보정계수, 언어 보정계수 산정 과정을 기술하였다. 또한 보정계수 산정 후, 좀더 깊이 있는 분석을 위해서는 대안들 사이에 트레이드오프를 식별하고, 전체적인 의사결정 과정에서 가장 중요한 판단기준을 찾고, 판단 기준의 가중치 변화에 따른 민감도를 평가하는 것이 필요하다[6]. 따라서, 민감도 분석을 통해 판단기준의 조정에 따른 보정계수 변화를 조사하였다. 먼저 어플리케이션 유형 보정계수의 판단기준으로 적용된 제어 복잡도, 데이터관리 복잡도, 처리 복잡도에 대한 가중치 값을 조정한 후, 어플리케이션 유형 보정계수에 대한 민감도를 분석한다. 다음으로 언어 보정계수의 판단기준으로 적용된 개발인력의 보편성, 개발코딩 시간, 디버깅 시간에 대한 가중치 값을 조정한 후 언어 보정계수의 민감도를 분석한다. 즉, 각 판단기준의 가중치를 조정함으로써 보정계수에 어떤 영향을 미치는지 민감도 분석을 통하여 조사하였다.

본 논문의 구성은 다음과 같다. 먼저 2장에서는 배경 지식으로서 보정계수 적용과 소프트웨어 개발비 대가기준 및 보정계수에 관하여 기술하고, 3장에서는 AHP 기법과 AHP 기법을 적용한 어플리케이션 유형 보정계수와 언어 보정계수 산정과정을 설명하고, 4장에서는 어플리케이션 유형 보정계수 산정에 적용된 3가지 판단기준과 언어 보정계수 산정에 적용된 3가지 판단기준 조정에 따른 민감도를 분석한다. 마지막으로 결론 및 향후 연구과제에 관하여 기술한다.

2. 관련 연구 및 연구배경

2.1 비용산정 모델의 보정 계수

상용 소프트웨어 산정 분야에서 거시적인 산정도구(macroestimation tools)와 미시적인 산정도구(microestimation tools)에서 산정 값을 보정하는 방식은 다소 상이하다[7].

거시적인 산정도구는 전형적으로 기술 수준, 도구, 기법과 같은 다양한 조정 요인으로부터 정보를 축적하여 전체 프로젝트에 대한 승수로 사용한다. 예를 들어, 만일 보통의 도구와 기법을 이용하는 평균 수준 팀의 월간 생산성이 PM당 10FP 혹은 1000LOC라면 우수한 도구와 기법을 갖춘 상위 수준 팀의 생산성은 2배인 PM당 20FP 혹은 2000LOC가 될 것이다. 역으로 그저 그런 도구와 기법을 갖춘 하위 수준 팀의 생산성은 PM당 5FP 혹은 500LOC에 불과할 것이다. 이러한 상황에서 팀의 능력과 개발 도구의 수준은 0.5에서

〈표 1〉 거시적인 산정 도구의 보정 예

요인들의 조합	승수
우수한 도구를 가진 상위 수준 팀	2.00
보통의 도구를 가진 상위 수준 팀	1.75
우수한 도구를 가진 평균 수준 팀	1.50
그저 그런 도구를 가진 상위 수준 팀	1.25
보통의 도구를 가진 평균 수준 팀	1.00
그저 그런 도구를 가진 평균 수준 팀	0.90
우수한 도구를 가진 하위 수준 팀	0.75
보통의 도구를 가진 하위 수준 팀	0.65
그저 그런 도구를 가진 하위 수준 팀	0.50

〈표 2〉 미시적인 산정 도구의 보정 예

등급	팀 승수	도구 승수
매우 우수	1.50	1.20
좋음	1.25	1.10
보통	1.00	1.00
낮음	0.75	0.90
매우 낮음	0.50	0.80

2.0사이의 생산성 승수로 변환된다. 비용 산정 도구 내부에 <표 1>과 유사한 테이블이 있을 수 있다.

이러한 9개의 조합이 어느 특정 산정 도구에서 얻어진 것은 아니지만, 이것은 다양한 요인들의 조합이 테이블 검색 기능에 의해 생산성 보정으로 변환되는 방식을 보여준다. 보통의 도구를 가진 평균 수준 팀이 PM당 10FP의 생산성을 가정하면, 우수한 도구를 가진 상위 수준 팀은 PM당 20FP의 생산성을 가정할 수 있을 것이고, 반면에 그저 그런 도구를 가진 하위 수준 팀은 PM당 5FP에 불과한 생산성을 가정할 수 있을 것이다.

예시한 것과 같이 이 두 가지 요인의 9가지 조합은 평균 중심값을 생성하고, 평균에 비해 양호하거나 양호하지 않은 상황을 다루기 위해 평균을 중심으로 양쪽 방향으로 4가지 조합을 산출한다.

미시적인 산정도구에서도 마찬가지로 보정 요인을 다루지만, 여러 요인들을 동시에 커버하는 집합적인 보정 요인을 이용하는 것이 아니라, <표 2>와 같이 전형적으로 각 요인을 독립적으로 적용한다. 각 요인을 개별적으로 적용하는 미시적인 산정 방식은 상황에 따라 특정 요인을 포함하거나 배제한다. 이렇게 하는 것이 매우 유용한 이유는 소프트웨어 생산성이라는 것은 소프트웨어 프로젝트의 결과에 영향을 미칠 수 있는 최소한 100개 이상의 요인들을 가진 복잡한 현상이기 때문이다.

더욱이 각 요인을 독립적으로 적용하는 것은 각 보정 요인의 가능한 범위를 제시하는 것과 동일한 의미이다. 예를 들어, 보통 수준의 도구를 가진 평균 수준 팀이 PM당 10FP의 생산성을 가진다고 가정하고 등급별 팀 승수와 도구 승수를 적용한 생산성은 아래의 <표 3>과 같은 조합이 얻어질 수 있다.

각 요인을 순차적으로 적용하는 이러한 방식은 무한히 계속 될 수 있다. 그러나, 만일 한 요인이 보통이고 곱하는 효과가 1이라고 가정하면, 프로젝트가 보통보다 양호하거나 양호하지 못한 요인인 경우에만 보정할 필요가 있다.

〈표 3〉 미시적인 산정 도구에서 보정 계수 적용 예

팀과 도구의 조합	생산성
매우 우수한 팀과 도구	10×1.5×1.2 = 18 FP/PM
매우 우수한 팀과 매우 낮은 도구	10×1.5×0.8 = 12 FP/PM
매우 낮은 팀과 매우 우수한 도구	10×0.5×1.2 = 6 FP/PM
매우 낮은 팀과 매우 낮은 도구	10×0.5×0.8 = 4 FP/PM

실제로 소프트웨어 프로젝트 결과에 영향을 줄 수 있는 100개 이상의 요인들이 존재하기 때문에, 보정 범위가 매우 크다는 것을 알 수 있다. SPR 지식 베이스에 의하면 대략 8000개의 소프트웨어 프로젝트가 PM당 0.13FP에서부터 PM당 140FP에 달한다. 즉, 소프트웨어 생산성의 범위는 매우 광범위하다. 그렇다면 여기에서 평균 생산성보다 매우 높거나 낮은 원인에 관한 의문이 제기될 수 있다.

소프트웨어 생산성에 영향을 미치는 요인 중에서 일부는 소프트웨어 프로젝트 팀의 통제 밖에 있다. 비록 이러한 요인들이 중요하더라도, 그것들을 조정할 수 있는 능력은 거의 없다. 예를 들어,

- 규모가 10,000FP 이상의 대형 시스템이 규모가 100FP 미만의 소형 프로젝트보다 생산성이 낮다.
- 미국의 경우, DoD 2167과 같은 표준을 준수하는 국방 소프트웨어 프로젝트는 방대한 양의 문서 작업을 요구하기 때문에 유사한 민간 프로젝트보다 생산성이 낮다.

여기에서 소프트웨어 개발팀의 통제 밖에 있는 요인들을 논의하는 것은 가치가 없다. 대신 관심의 대상이 되는 것은 도구의 선택, 프로그래밍 언어, 개발 공정과 같은 통제 수단이 존재하는 요인들이다.

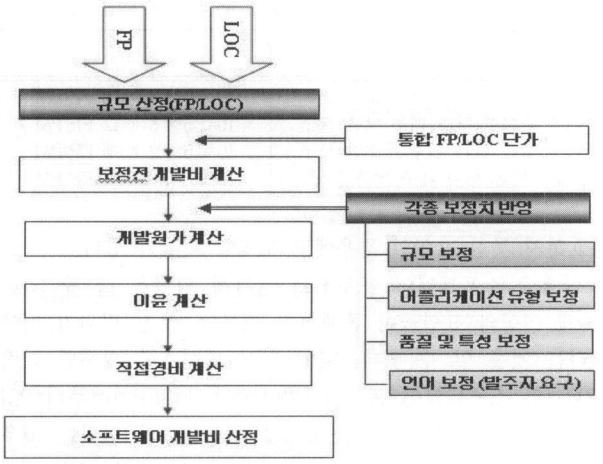
참고적으로 대표적인 소프트웨어 산정 도구인 COCOMO 모델에서는 델파이 기법을 적용하여 보정계수를 산정하였으며[8], 일반적인 상용 소프트웨어 산정 도구에서는 모델의 알고리즘 및 보정계수 유도과정을 공개하지 않고 있다.

2.2 소프트웨어 개발비 대가기준의 보정계수

소프트웨어 개발비 대가기준은 국가기관 등이 소프트웨어 개발, 데이터베이스 구축, 정보전략 계획 수립 등의 정보화 사업을 추진함에 있어 정보통신기술의 발전 및 사회적 여건 변화에 유연하게 대처하고, 소프트웨어 산업과의 선순환적 구조를 가질 수 있도록 소프트웨어사업에 대한 예산수립이나 발주 시 적정비용 등을 산정하기 위한 기준을 제공한다.

소프트웨어 개발비 대가기준에서 보정계수는 개발 규모에 의한 산정방법 시 개발 대상 소프트웨어의 규모를 기능점수 또는 LOC 등의 적정한 방식에 따라 산정하고 개발 및 적용 환경의 특성에 따라 이를 보정하여 산정하는 방식으로 산정 절차는 (그림 1)과 같다[4].

(그림 1)에서처럼 현행 대가기준에서는 규모 보정, 어플리케이션 유형 보정, 품질 및 특성 보정, 언어 보정의 4가지 보정계수를 적용하며 그 중 본 논문에서 다루고자 하는 것



(그림 1) 소프트웨어 비용산정 과정

은 어플리케이션 유형 보정계수와 언어 보정계수이다.

3. 보정계수 유도

3.1 AHP 기법

1970년대 초반 T. Saaty에 의하여 개발된 계층분석적 의사결정방법(Analytic Hierarchy Process : AHP)은 의사결정의 계층구조를 구성하고 있는 요소간의 쌍대비교(Pairwise comparison)에 의한 판단을 통하여 평가자의 지식, 경험 및 직관을 포착하고자 하는 의사결정방법론이다. 일반적으로 의사결정문제는 불완전한 정보 및 제한된 자원 하에서 최적의 대안을 선택해야 하는 문제를 내포하고 있다. AHP는 이러한 다수 기준 하에서 평가되는 다수 대안들의 우선순위를 선정하는 문제를 다루며, 기존의 의사결정이론 체계에서 보면 다속성 의사결정분석(multi-attribute decision making)의 선호보정이 있는 모형(compensatory preference model)으로서 그 속성을 위치시킬 수 있다. 적용 절차는 다음과 같다.

[적용절차]

step 1 : 의사결정 문제를 상호관련 된 의사결정 사항들의 계층으로 분류하여 의사결정계층(decision hierarchy)을 설정한다.

AHP 적용에서 가장 중요한 단계로 계층의 최상위층에는 가장 포괄적인 의사결정의 목적이 놓여지며, 그 다음의 계층들은 의사결정의 목적에 영향을 미치는 다양한 요소들로 구성된다.

step 2 : 의사결정 요소들 간의 쌍대비교로 판단자료를 수집한다.

상위계층에 있는 요소들의 목표를 달성하는데 공헌하는 직계하위계층에 있는 요소들을 쌍대비교 하여 행렬을 작성한다. 작성된 쌍대비교행렬 A는 다음과 같이 행렬의 대각을 중심으로 역수의 형태를 취하게 된다. 아래의 식(1)과 같다.

$$A=\begin{bmatrix} 1 & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & 1 & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & 1 & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & 1 \end{bmatrix} \quad (1)$$

여기서, $a_{ij}=1/a_{ji},\ a_{ii}=1,\ \forall i$

AHP에서의 판단자료는 계층 내 요소간의 쌍대비교를 통하여 도출한 요소간의 상대적 중요도를 나타내는 점 추정치를 사용하는데, 쌍대비교를 통한 정량적인 판단을 수행하기 위해서는 신뢰할만하고 이용 가능한 척도가 필요하며, 이를 위하여 통상 9점 척도가 많이 이용되고 있다.

step 3 : 고유치방법을 사용하여 의사결정요소들의 상대적인 가중치를 추정한다.

한 계층 내에서 비교 대상이 되는 n개 요소의 상대적인 중요도를 $w_i\,(i=1,\cdots,n)$라 하며, 상기한 쌍대비교행렬에서의 a_{ij}는 $w_i/w_j(i,j=1,\ \cdots,n)$로 추정할 수 있다. 즉, a_{ij}와 w_i 사이에는 다음 식(2)가 성립한다.

$$a_{ij}=w_i/w_j \qquad (i,j=1,\ \cdots,n) \quad (2)$$

여기서, 행렬의 모든 요소를 나타내면 다음 식 (3)과 같다.

$$\sum_j^n a_{ij}\cdot w_j\cdot\frac{1}{w_i}=n \quad (i,j=1,\ .....,n) \quad (3)$$

위 식(3)은 다음 식(4)와 같이 나타낼 수 있고,

$$\sum_j^n a_{ij}\cdot w_j=n\cdot w_i \qquad (i,j=1,\ ....,n) \quad (4)$$

위 식 (4)는 선형대수론에서의 고유치 문제와 같다. 즉, 요소 a_{ij}로 구성되는 행렬 A를 다음 식 (5)와 같이 나타낼 때,

$$A=\begin{bmatrix} w_1/w_1 & w_1/w_2 & w_1/w_3 & \ldots & w_1/w_n \\ w_2/w_1 & w_2/w_2 & w_2/w_3 & \ldots & w_2/w_n \\ \cdot & \cdot & \cdot & \ldots & \cdot \\ w_n/w_1 & w_n/w_2 & w_n/w_3 & \ldots & w_n/w_n \end{bmatrix} \quad (5)$$

고유치방법에 의하여 식 (6)과 같이 나타낼 수 있다.

$$A\cdot w=n\cdot w \quad (6)$$

$w=[w_1, w_2, w_3, ..., w_n]$: 행렬 A의 우측 고유벡터

n : 행렬 A의 고유치

가중치 추정을 위한 방법으로 고유치 방법이 널리 사용되고 있으나, 현재까지 제안된 방법으로는 이외에도 산술평균, 기하평균, 최소자승법, 조화평균, 평균치환법 등이 있으며 이

에 관한 계속적인 연구가 진행되고 있다.

step 4 : 평가대상이 되는 여러 대안들에 대한 종합순위를 얻기 위하여 의사결정 요소들의 상대적인 가중치를 종합화한다.

계층의 최상위에 있는 대안들의 우선순위를 결정하는 종합중요도벡터를 산출하는데, 이는 step 3에서 구한 각 계층에서의 가중치를 종합함으로써 가능하다. 구체적으로 최상위 계층에 대하여 k번째 하위계층에 있는 대안들의 종합중요도는 다음 식 (7)을 통하여 구할 수 있다.

$$C[1,k] = \prod_{i=2}^{k} B_i \qquad (7)$$

$C[1,k]$: 첫 번째 계층에 대한 k번째 계층요소의 종합가중치

B_i : 추정된 w벡터를 구성하는 행을 포함하는 $n_{i-1} \cdot n_i$ 행렬

n_i : i 번째 계층의 요소 수

전체 계층의 종합중요도를 최하위 계층에 대한 직계 상위계층의 가중치행렬에 그 상위계층에서 구한 가중치행렬을 곱하고 이 과정을 상위계층으로 반복하여 구하는 방법은 논란의 여지가 없이 인정되고 있다[9].

3.2 AHP 기법을 이용한 보정계수 산정

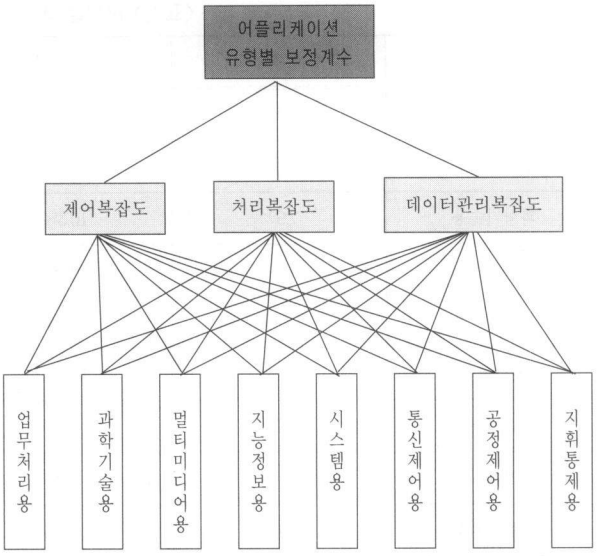
AHP 기법을 적용하여 어플리케이션 유형 보정계수와 언어 보정계수를 산정하였다.

3.2.1 어플리케이션 유형 보정계수

어플리케이션 유형별 보정계수를 AHP 기법으로 산정하기 위하여 의사결정계층을 다음 (그림 2)와 같이 구성하였다.

목표 레벨로 어플리케이션 유형별 보정계수를 산정하는 것으로 삼았으며, 각 대안들은 업무처리용, 과학기술용, 멀티미디어용, 지능정보용, 시스템용, 통신제어용, 공정제어용, 지휘통제용으로 구분되며 이들은 제어 복잡도, 데이터관리 복잡도, 처리 복잡도의 판단기준으로 평가하는 의사결정 계층도를 구성하였다.

판단기준을 제어 복잡도, 데이터관리 복잡도, 처리 복잡도로 분류한 근거는 다음과 같다. Price-S에서는 어플리케이션 내의 복잡



(그림 2) 어플리케이션 유형별 보정계수를 위한 계층도

도를 명령과 제어(Command & control), 통신(Communication), 데이터 저장과 추출(Store & retrieve data), 문자열조작(String manipulation), 수학연산(Mathematical operations), 운영체제(Operating systems), 상호작용연산(Interactive operations)의 7가지 요소로 구분하고 있다(Price systems). 또한 COCOMO II에서도 어플리케이션의 복잡도를 평가하는 기준을 제어연산(Control operation), 계산연산(Computational operation), 장치종속연산(Device-dependent operation), 데이터 관리연산(Data management operation), 사용자 인터페이스 관리연산(User interface management operation)등의 5가지 요소로 구분하고 있다(Boehm, 2000). AHP 분석의 용이성을 위해 위 요소를 담고 있는 제어 복잡도, 데이터관리 복잡도, 처리 복잡도의 3가지로 분류하였다. 제어 복잡도는 어플리케이션 구성요소간의 통신과 제어(communication & control)가 엄격한 시간제한(time constraint) 조건에서 이루어지는 경우 소프트웨어 개발비용은 증가한다. 즉, 시간제한의 엄격성이 높은 실시간 시스템의 경우 개발비용이 증가한다. 데이터관리 복잡도는 어플리케이션의 데이터 관리 기능이 많으면 소프트웨어 개발비용은 증가한다. 즉, 관리해야 할 데이터의 양과 기능이 많을수록 소프트웨어 개발비용은 증가한다. 처리 복잡도는 어플리케이션의 처리(processing) 알고리즘이

〈표 4〉 어플리케이션 유형 판단기준

판단기준	PRICE-S	COCOMO II
제어 복잡도	- 명령과 제어 - 통신 - 운영체제	- 제어연산 - 장치종속연산
데이터관리 복잡도	- 데이터 저장과 추출 - 상호작용연산	- 데이터관리연산 - 사용자인터페이스 관리 연산
처리 복잡도	- 문자열 조작 - 수학연산	- 계산연산

〈표 5〉 AHP를 이용한 어플리케이션 유형 보정계수

	제어 복잡도	데이터 관리복잡도	처리 복잡도	상대적 가중치	보정계수
상대적 가중치	30.8%	36.1%	33.1%		
업무처리용	5.1%	10.2%	7.2%	7.6%	1.0
과학기술용	7.9%	10.3%	10.0%	9.5%	1.3
멀티미디어용	9.5%	10.7%	10.1%	10.1%	1.3
지능정보용	12.8%	13.5%	13.1%	13.2%	1.7
시스템용	13.5%	12.2%	13.6%	13.1%	1.7
통신제어용	16.0%	13.1%	14.3%	14.4%	1.9
공정제어용	16.5%	13.9%	14.9%	15.0%	2.0
지휘통제용	18.6%	16.1%	16.8%	17.1%	2.3

〈표 6〉 AHP를 이용한 개발언어 보정계수

	개발인력의 보 편 성	개발 코딩시간	디버깅 시간	상대적 가중치	보정계수
상대적 가중치	33.1%	39.4%	27.5%		
1세대 언어	47.0%	40.3%	39.5%	42.3%	1.5
2·3세대 언어	24.1%	29.3%	28.6%	27.4%	1.0
4세대 언어	14.8%	19.1%	19.6%	17.8%	0.6
5세대 언어	14.1%	11.4%	12.3%	12.5%	0.5

복잡할수록 소프트웨어 개발비용은 증가한다. 이들의 대응 관계는 <표 4>와 같다[5, 10].

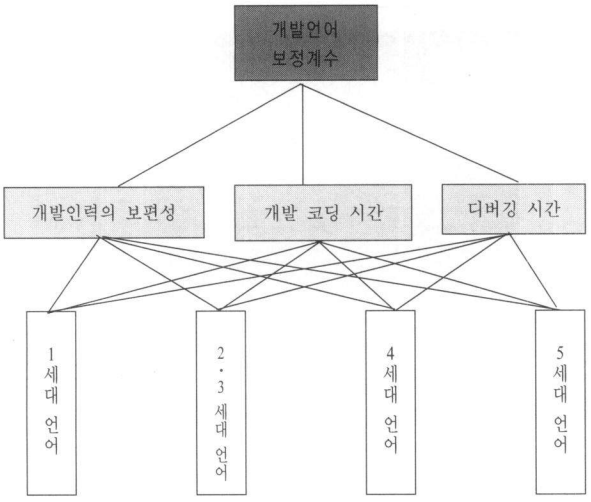
위 (그림 2)와 같이 의사결정 계층도를 구성하고 평가 기준인 제어 복잡도, 처리 복잡도, 데이터관리 복잡도에 대한 상대적 가중치를 평가한 설문으로 분석하였다. 1차적으로 회수된 145개의 데이터 중 2개의 잘못된 데이터와 1개의 이상치 데이터를 제외한 142개의 유효데이터로 각 평가 기준에 대한 상대적 가중치를 평균으로 산정하였다. 다음으로 제어 복잡도, 처리 복잡도, 데이터관리 복잡도 각각에 대한 어플리케이션 유형별 일대일비교표를 작성하여 각 기준에 대한 어플리케이션 유형별 상대적 가중치를 조사하였다[11]. [11]은 소프트웨어 사업대가기준 개정 작업의 일환으로 수행되었다. 이를 근거로 산출된 각 판단기준 및 대안에 대한 상대적 가중치 값을 얻은 후, 업무처리용의 상대적 가중치 값인 '7.6%'를 기준 '1.0'으로 삼아 나머지 7가지 유형인 과학기술용, 멀티미디어용, 지능정보용, 시스템용, 통신제어용, 공정제어용, 지휘통제용의 비례 값을 보정계수로 산정하였다. 그 결과 상대적 가중치 및 보정계수는 <표 5>와 같다.

3.2.2 언어 보정계수

언어 유형별 보정계수를 AHP 기법에 의하여 산정하기 위하여 의사결정계층을 다음 (그림 3)과 같이 구성하였다. 즉, 목표 레벨로 언어 보정계수를 산정하는 것으로 삼았으며, 각 대안들은 1세대 언어, 2·3세대 언어, 4세대 언어, 5세대 언어로 구분되며, 이들은 개발인력의 보편성과 개발코딩 시간, 디버깅 시간을 판단 기준으로 평가하는 의사결정 계층도를 구성하였다.

언어 측면에서는 언어에 대한 인력 측면과 생산성 측면이 주요인이다. 따라서 판단기준을 개발언어의 인력 측면에서 개발인력의 보편성을 선정하였으며, 이는 각 개발언어별로 소프트웨어 인력시장에서 개발인력의 희소성의 정도를 의미한다. 언어에 대한 개발인력의 보편성이 낮을수록 인력을 구하기 힘들게 되므로 소프트웨어 개발 비용을 증가하는 관계를 판단 기준으로 설정하였다. 다음으로 개발 언어의 생산성 측면으로 언어에 대한 개발코딩 시간과 디버깅 시간을 판단기준으로 선정하였다. 개발코딩 시간과 디버깅 시간은 개발의 어려움 정도를 의미하는 것으로 개발코딩 시간이 길어질수록, 디버깅 시간에 소요되는 시간이 길어질수록 소프트웨어 개발비용은 증가한다는 것을 판단기준으로 설정하였다[5, 10, 11].

위 (그림 3)과 같이 의사결정 계층도를 구성하고 평가기준인 개발인력의 보편성, 개발 코딩 시간, 디버깅 시간에 대한 상대적 가중치를 평가한 설문으로 분석하였다[11].설문 데이터를 근거로 산출된 각 판단기준 및 대안에 대한 상대적 가중치 값과 보정계수는 <표 6>과 같다.



(그림 3) 개발언어 보정계수를 위한 계층도

4. 민감도 분석

민감도 분석은 입력변수들과 설정된 모델 출력변수간의 불확실성을 평가하는 일종의 확률론적인 방법으로 모델이 입력변수들의 변화에 어떻게 영향을 받는지 평가하여 모델의 신뢰성과 예측 결과에 대한 신뢰성을 향상시키기 위해 사용된다. 따라서, 민감도 분석은 모델 입력변수의 불확실성의 결과로서 얻어지는 전체 불확실성을 정량화하는데 목적을 둔 불확실성 해석과 밀접한 관계가 있다[12].

의사결정문제와 관련된 정보의 변화에 따른 민감도를 분석하기 위해 AHP의 민감도 분석을 활용하여 판단기준의 가중치를 변화시켜 나감으로써 대안의 우선순위가 어떻게 변화하는지를 검토할 수 있다. 이러한 민감도 분석에는 성과 민감도, 동적 민감도, 경사 민감도, 2차원 구성, 가중치차이 민감도 등 다섯 가지 방법이 제공되고 있다[9].

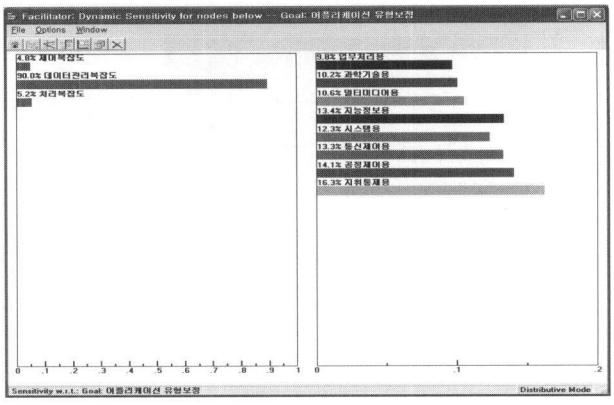
본 연구에서는 동적 민감도를 이용하여 어플리케이션 유형 보정계수의 판단기준인 제어 복잡도, 데이터관리 복잡도, 처리 복잡도 각각에 대한 중요도를 조정할 때 나머지 판단기준의 중요도에 미치는 영향과 8가지 어플리케이션 유형 보정계수에 미치는 영향을 조사하였다. 마찬가지로 언어 보정계수의 판단기준인 개발인력의 보편성, 개발코딩 시간, 디버깅 시간 각각에 대한 중요도를 조정할 때 나머지 판단기준에 미치는 영향과 개발언어 보정계수에 미치는 영향을 조사하였다. 다음으로 하나의 기준이 갖는 우선순위의 변화에 따라 대안이 갖는 우선순위의 변화를 표현하는 경사민감도(Gradient Sensitivity)를 조사하였다.

4.1 판단기준의 가중치 조정에 따른 동적 민감도 분석

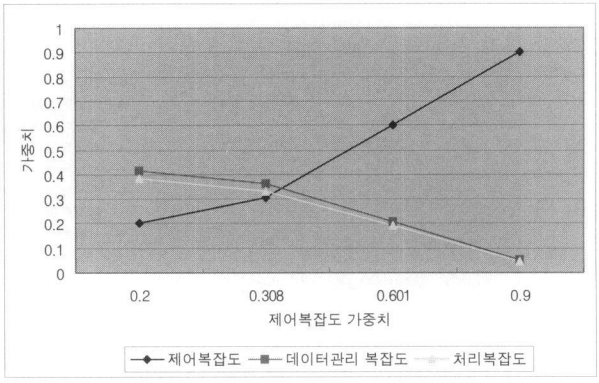
각 판단기준의 가중치 변화에 따른 보정계수의 민감도를 분석하였다.

기준을 나타내는 막대의 길이를 변화시킴으로써 대안의 우선 순위 가중치를 나타내는 막대의 길이가 변화하게 된다.

아래의 (그림 4)에서 왼쪽에 위치한 막대 그래프는 어플리케이션 유형 보정계수의 판단기준인 제어 복잡도, 데이터관리 복잡도, 처리 복잡도를 나타낸다. 이 3가지 판단기준의 가중치를 (그림 4)와 같은 동적 민감도를 이용하여 조정하였다. 하나의 판단기준 가중치를 조정했을 때 나머지 2개의 판단기준에는 어떤 영향을 미치는가를 살피고, 보정계수와는 어떤 관계가 있는지 조사하였다.



(그림 4) 어플리케이션 유형 보정계수의 동적민감도



(그림 5) 제어복잡도의 가중치 조정

4.1.1 어플리케이션 유형 보정계수

제어복잡도의 가중치를 줄이거나 늘렸을 때, 나머지 2가지 판단기준의 가중치는 (그림 5)와 같은 유형을 보였다.

나머지 판단기준인 데이터관리 복잡도, 처리 복잡도에 대해서도 (그림 5)와 비슷한 유형을 보였다.

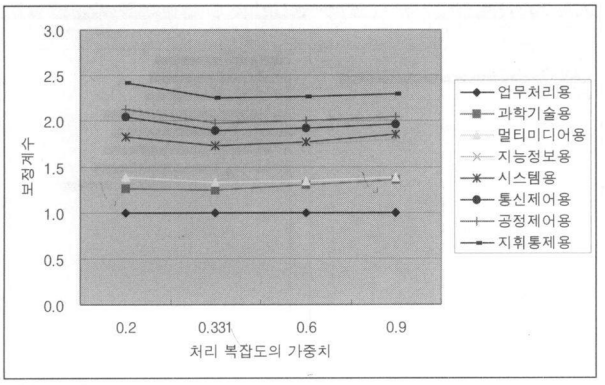
다음으로 판단기준의 가중치를 조정하였을 때, 각 보정계수에 미치는 영향을 조사한 결과는 아래의 (그림 6), (그림 7), (그림 8)과 같다.

(그림 6)은 처리 복잡도 가중치 조정에 따른 어플리케이션 유형 보정계수 값의 변화를 나타낸다.

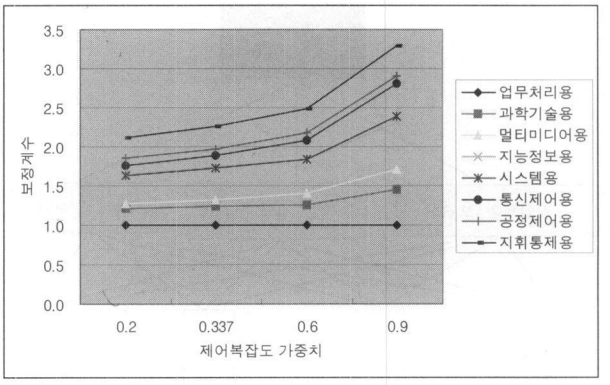
(그림 6)에서 알 수 있듯이 처리 복잡도 가중치 값이 증가될수록 전체적인 보정계수 값은 낮아짐을 보인다.

(그림 7)은 데이터관리 복잡도 가중치 조정에 따른 어플리케이션 유형 보정계수 값의 변화를 나타낸다.

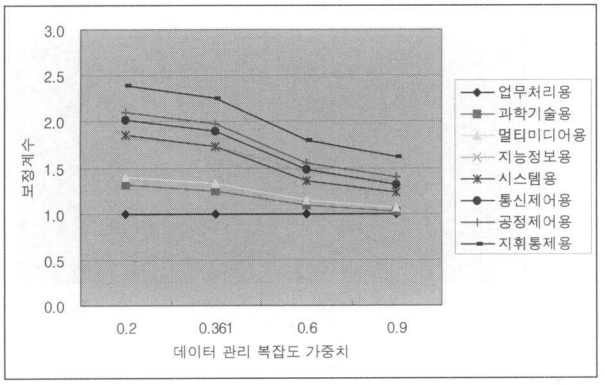
(그림 7)에서 알 수 있듯이 데이터관리 복잡도 가중치 값이 증가될수록 전체적인 보정계수 값은 감소됨을 보인다.



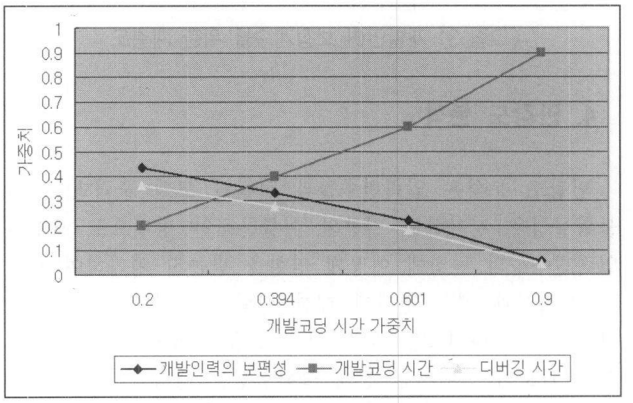
(그림 6) 처리 복잡도 가중치 조정에 따른 보정계수 변화



(그림 8) 제어 복잡도 가중치 조정에 따른 보정계수 변화



(그림 7) 데이터관리 복잡도 가중치 조정에 따른 보정계수 변화



(그림 9) 개발코딩 시간의 가중치 조정

특히, 지휘통제용의 변화가 크게 나타난다.

(그림 8)은 제어 복잡도 가중치 조정에 따른 어플리케이션 유형 보정계수 값의 변화를 나타낸다.

(그림 8)에서 알 수 있듯이 제어 복잡도 가중치 값이 증가될수록 전체 보정계수 값은 크게 증가됨을 보인다.

위의 4가지 그래프에서 알 수 있듯이 하나의 판단기준의 가중치 값을 조정하였을 때 나머지 판단기준의 가중치에 대한 변화는 비슷한 유형을 보였다. 그러나, 8가지 어플리케이션 유형의 보정계수에 적용시켰을 때에는 처리 복잡도와 데이터관리 복잡도의 가중치를 증가시켰을 때, 보정계수 값은 대체적으로 낮아짐을 보인다. 특히 데이터관리 복잡도의 가중치가 증가될수록 보정계수 값은 더 낮아진다. 그러나, 제어 복잡도 가중치를 증가시켰을 때, 전체적인 보정계수 값은 증가됨을 보인다. 이것으로 알 수 있듯이 어플리케이션 유형 보정계수 값은 처리 복잡도에 비해 데이터관리 복잡도와 제어 복잡도에 민감하게 반응하며 특히 데이터관리 복잡도에 대해서는 보정계수 값이 낮아지는 반면, 제어 복잡도에 대해서는 보정계수 값이 높아지는 것을 알 수 있다.

4.1.2 언어 보정 계수

개발코딩 시간의 가중치를 줄이거나 늘렸을 때, 나머지 2가지 판단기준의 가중치는 (그림 9)와 같은 유형을 보였다.

나머지 판단기준인 개발인력의 보편성과 디버깅 시간에 대해서도 (그림 9)와 비슷한 유형을 보였다.

다음으로 판단기준의 가중치를 조정하였을 때, 각 보정계수에 미치는 영향을 조사한 결과는 아래의 (그림 10), (그림 11), (그림 12)와 같다.

(그림 10)은 개발인력의 보편성 가중치 조정에 따른 언어 보정계수 변화를 나타낸다.

(그림 10)에서 알 수 있듯이 개발인력의 보편성 가중치 값이 증가할수록 1세대 언어와 5세대 언어의 보정계수 값은 증가됨을 보인다.

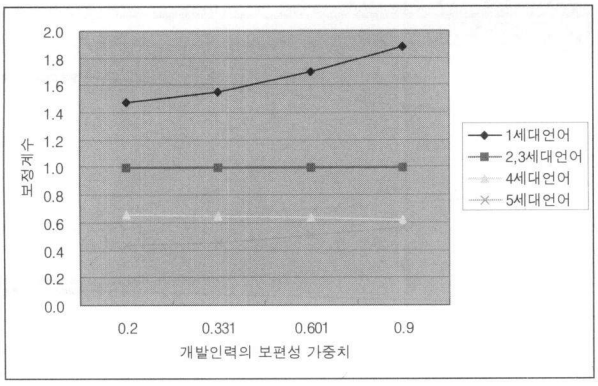
(그림 11)은 개발코딩 시간 가중치 조정에 따른 개발언어 보정계수의 변화를 나타낸다.

(그림 11)에서 알 수 있듯이 개발코딩 시간의 가중치 값이 증가할수록 1세대언어와 5세대 언어의 보정계수 값은 점점 낮아지며 4세대언어는 거의 변화를 보이지 않는다.

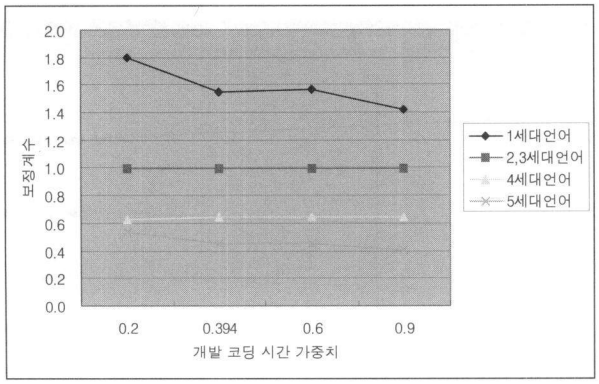
(그림 12)는 디버깅 시간 가중치 조정에 따른 언어 보정계수 변화를 나타낸다.

(그림 12)에서 알 수 있듯이 디버깅 시간 가중치 값이 증가할수록 언어 보정계수 값은 1세대 언어를 제외하고 나머지 언어에 대해서는 거의 비슷한 값을 유지하는 것으로 나타났다.

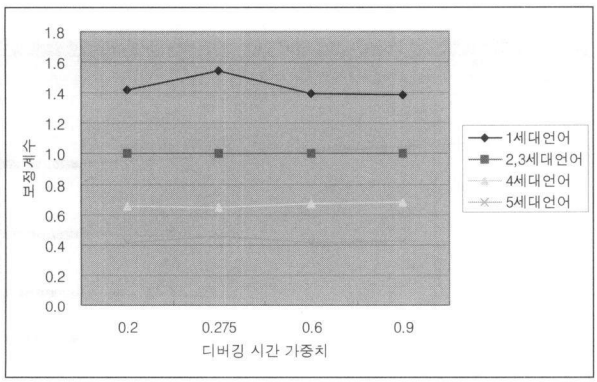
위의 4가지 그래프에서 알 수 있듯이 하나의 판단기준의 가중치 값을 조정하였을 때 나머지 판단기준 가중치에 대한 변화는 비슷한 유형을 보였다. 그러나, 4가지 언어 보정계수



(그림 10) 개발인력의 보편성 가중치 조정에 따른 보정계수 변화

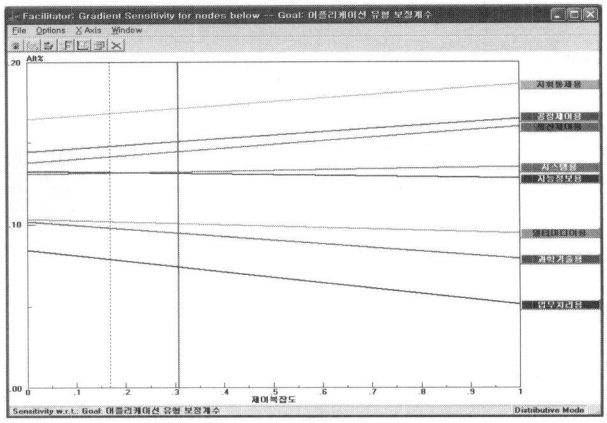


(그림 11) 개발코딩 시간 가중치 조정에 따른 보정계수 변화



(그림 12) 디버깅 시간 가중치 조정에 따른 보정계수 변화

에 적용시켰을 때에는 개발코딩 시간과 디버깅 시간은 가중치를 증가시켰을 때, 언어 보정계수 값은 낮아짐을 보인다. 개발코딩 시간에서의 변화율이 조금 더 낮아진다. 그러나, 개발인력의 보편성 가중치를 증가시켰을 때, 대체적으로 보정계수 값은 증가됨을 보인다. 이것으로 알 수 있듯이 언어 보정계수 값은 디버깅 시간에 비해 개발인력의 보편성과 개발코딩 시간에 더 민감하게 반응하며 특히 개발코딩 시간에 대해서는 보정계수 값이 낮아지는 반면, 개발인력의 보편성에 대해서는 보정계수 값이 높아지는 것을 알 수 있다.



(그림 13) 제어 복잡도 가중치에 따른 경사민감도

4.2 판단기준의 경사 민감도 분석

경사 민감도는 하나의 기준이 갖는 가중치 값이 변화함에 따라 대안이 갖는 우선순위의 변화를 표현하는 분석 방법이다. 경사민감도를 사용하여 제어복잡도, 데이터 관리 복잡도, 처리 복잡도에 따른 어플리케이션 유형의 우선순위 가중치가 어떻게 변화하는지를 보이며, 마찬가지로 개발인력의 보편성, 개발코딩 시간, 디버깅 시간의 변화에 따라 언어 별 우선순위 가중치가 어떻게 변화하는지 조사할 것이다.

4.2.1 어플리케이션 유형의 경사 민감도

아래의 (그림 13)은 제어 복잡도 가중치에 따른 각 보정계수의 경사민감도를 나타낸다.

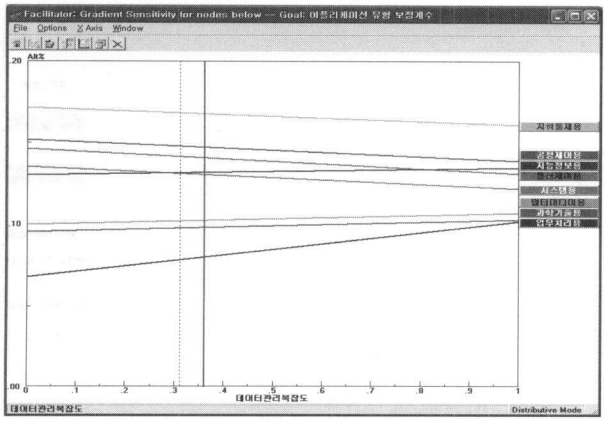
(그림 13)에서 알 수 있듯이 제어복잡도 가중치가 0.17이상인 지점에서부터 시스템용과 지능정보용의 우선순위는 시스템용이 더 높은 것으로 나타났다. 제어복잡도의 가중치가 높아질수록 지휘통제용, 공정제어용, 통신제어용, 시스템용의 우선순위 가중치는 점점 높아지는 반면, 업무처리용, 과학기술용, 멀티미디어용, 지능정보용의 우선순위 가중치는 점점 낮아짐을 보인다.

(그림 14)는 데이터관리 복잡도 가중치에 따른 각 보정계수의 경사민감도를 나타낸다.

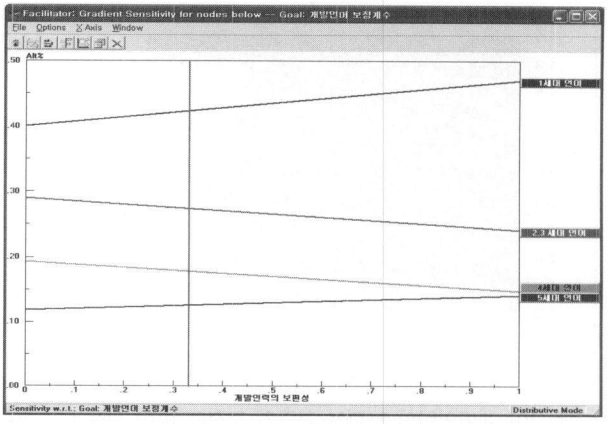
(그림 14)에서 알 수 있듯이 데이터관리 복잡도 가중치가 0.28이상인 지점에서부터 지능정보용이 시스템용의 우선순위보다 높게 나타나며, 0.83이상인 지점부터는 지능정보용이 통신제어용의 우선순위보다 더 높게 나타난다. 데이터관리 복잡도 가중치가 높아질수록 업무처리용, 과학기술용, 멀티미디어용, 지능정보용의 우선순위가중치는 점점 높아지는 반면 지휘통제용, 공정제어용, 통신제어용, 시스템용 가중치는 점점 낮아짐을 보인다.

(그림 15)는 처리 복잡도 가중치에 따른 각 보정계수의 경사민감도를 나타낸다.

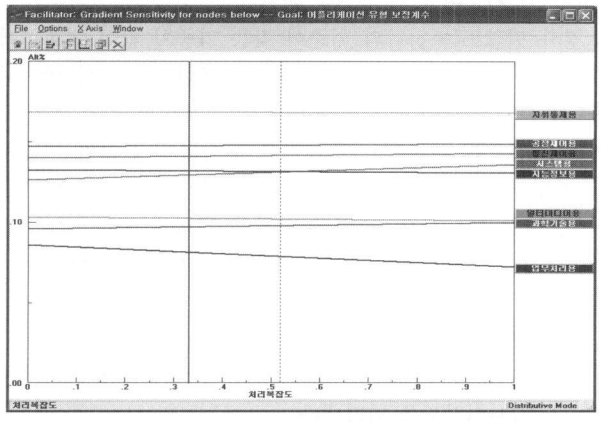
(그림 15)에서 알 수 있듯이 처리 복잡도 가중치가 0.52이상인 지점에서부터 시스템용이 지능정보용보다 우선순위가 중치가 높게 나타난다. 지휘통제용은 처리 복잡도 가중치가 증가되어도 거의 영향을 받지 않는 것으로 나타났다. 처리 복잡도 가중치가 높아질수록 공정제어용, 통신제어용, 과학



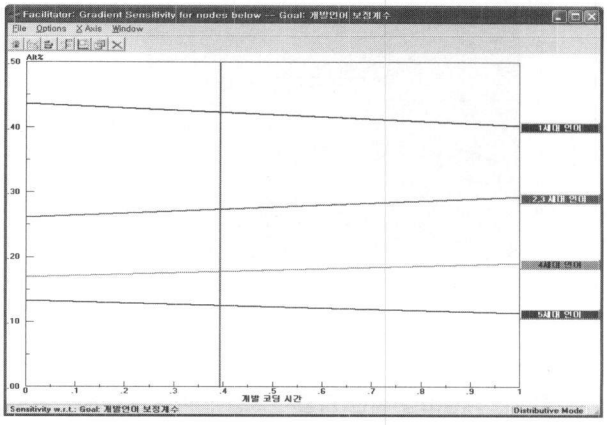
(그림 14) 데이터관리 복잡도 가중치에 따른 경사민감도



(그림 16) 개발인력의 보편성 가중치에 따른 경사민감도



(그림 15) 처리 복잡도에 따른 경사민감도



(그림 17) 개발코딩시간 가중치에 따른 경사민감도

기술용은 우선순위 가중치가 조금 증가됨을 보였고, 반면 지능정보용, 멀티미디어용, 업무처리용은 낮아짐을 보였다. 특히 업무처리용은 처리 복잡도에 가장 민감하게 반응하는 것으로 나타났다.

4.2.2 언어의 경사 민감도

아래의 (그림 16)은 개발인력의 보편성 가중치에 따른 언어 보정계수의 경사민감도를 나타낸다.

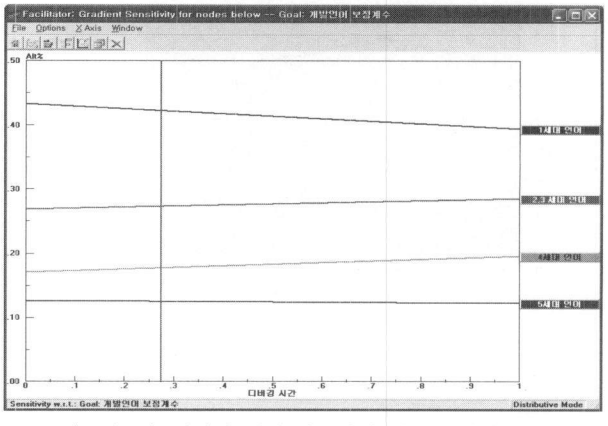
(그림 16)에서 알 수 있듯이 개발인력의 보편성 가중치가 증가할수록 1세대언어와 5세대언어의 우선순위는 높아진다. 반면에, 2·3세대언어와 4세대언어의 우선순위는 점점 낮아지고 있다.

(그림 17)은 개발코딩 시간 가중치에 따른 언어 보정계수의 경사민감도를 나타낸다.

(그림 17)에서 알 수 있듯이 개발코딩 시간 가중치가 증가할수록 2·3세대언어와 4세대언어의 우선순위는 조금씩 높아진다. 반면에 1세대언어와 5세대언어의 우선순위는 점점 낮아지고 있다.

(그림 18)은 디버깅 시간 가중치에 따른 언어 보정계수의 경사민감도를 나타낸다.

(그림 18)에서 알 수 있듯이 디버깅 시간 가중치가 증가할수록 2·3세대언어와 4세대언어의 우선순위는 조금씩 높아진다. 반면에 1세대언어와 5세대언어의 우선순위는 점점 낮아지고 있다. 특히, 1세대언어의 감소율이 크게 나타난다.



(그림 18) 디버깅 시간 가중치에 따른 경사민감도

경사 민감도를 이용하여 언어의 판단기준을 조사한 결과 개발인력의 보편성에 대한 중요도가 높아질수록 1세대, 5세대 언어의 우선순위 가중치가 높아짐을 보이고, 2·3세대, 4세대언어는 개발 코딩시간, 디버깅 시간의 중요도가 높아질수록 우선순위 가중치가 높아짐을 알 수 있다.

5. 결론 및 향후 연구과제

성공적인 소프트웨어 프로젝트 수행을 결정짓는 중요한 작업 중 하나는 소프트웨어 개발비용을 정확하게 산정하는 것이다. 정확한 산정을 위해서는 빠르게 변화하는 개발 환경 및 기술 변화에 따른 변경 요인을 비용 산정 과정 시 반영시켜야 하며 이를 위해서는 적절한 보정계수 선정과 보정계수 값 적용이 중요시된다.

이에 본 논문에서는 정보통신부 고시 소프트웨어 비용 산정 기준인 소프트웨어 개발비 대가기준을 위한 어플리케이션 유형 보정계수와 언어 보정계수 산정과정을 기술하였다. 또한, 민감도 분석을 통하여 각 판단기준의 가중치 조정에 따른 각각의 어플리케이션 유형 보정계수와 언어 보정계수에 어떠한 영향을 미치는지 조사하였다.

결론적으로, 어플리케이션 유형 보정계수는 처리 복잡도에 비해 데이터관리 복잡도와 제어 복잡도에 민감하게 반응하였다. 처리 복잡도 가중치 값을 0.331에서 0.6으로 증가시켰을 때 시스템용 어플리케이션 보정계수를 제외한 나머지 7가지 어플리케이션 보정계수에는 변화가 없었다. 반면, 데이터관리 복잡도 가중치 값이 0.361에서 0.6으로 증가될 때 지휘통제용 어플리케이션의 보정계수는 2.3에서 1.8로, 공정제어용 어플리케이션 보정계수는 2.0에서 1.6으로, 통신제어용 어플리케이션 보정계수는 1.9에서 1.5로 감소됨을 보였다. 또한 제어 복잡도 가중치 값을 0.337에서 0.6으로 증가시켰을 때 지휘통제용 어플리케이션의 보정계수는 2.3에서 2.5로, 공정제어용 어플리케이션 보정계수는 2.0에서 2.2으로, 통신제어용 어플리케이션 보정계수는 1.9에서 2.1로 증가됨을 보였다. 데이터관리 복잡도 가중치 값이 증가될수록 보정계수 값이 낮아지는 반면, 제어 복잡도 가중치 값이 증가될수록 보정계수 값이 높아짐을 알 수 있다. 따라서, 제어 복잡도 가중치가 높은 어플리케이션에 대해서는 어플리케이션 유형 보정계수 값의 상향 조정이 필요함을 알 수 있었다.

또한, 언어 보정계수 값은 디버깅 시간에 비해 개발인력의 보편성과 개발코딩 시간에 더 민감하게 반응하였다. 개발인력의 보편성 가중치 값이 0.331에서 0.6으로 증가시켰을 때 1세대 언어의 보정계수는 1.5에서 1.7로 증가됨을 보였다. 또한 개발코딩 시간 가중치 값에 대해서는 가중치 값이 0.394에서 0.2로 감소시켰을 때 1세대 언어의 보정계수는 1.5에서 1.8로 증가하였고, 가중치 값이 0.6이상인 지점부터 보정계수는 감소됨을 보였다. 언어 보정계수에서 개발코딩 시간의 가중치 값이 높아질수록 보정계수 값이 낮아지는 반면, 개발인력의 보편성 가중치 값이 높아질수록 1세대 언어와 5세대 언어의 보정계수 값은 높아짐을 알 수 있었다. 따라서, 개발인력의 보편성에 대한 중요도가 높아지는 언어에 대해서는 보정계수의 상향 조정이 필요함을 알 수 있었다.

향후, 연구과제로는 국내의 소프트웨어 개발 환경과 기술을 적절히 반영시킬 수 있는 추가적인 보정계수 선정과 적절한 보정계수 값 조정에 대하여 연구할 것이다. 또한, 언어 보정계수 구분 시 세대별 분류가 아닌, 구체적인 개발언어를 들어 좀더 세부적으로 구분한 후 보정계수를 산정하고자 한다.

참 고 문 헌

[1] T. Capers Jones, "How Software Estimation Tools Work," Version 5, 2005. 2.

[2] S.D. Conte et al.,"Software Engineering Metrics and Models," The Benjamin/Cummings Publishing Company, Inc. 1985.

[3] Lionel C. Briand and Isabella Wieczoek, "Resource Estimation in Software Engineering," ISERN, 2000. 05.

[4] 한국소프트웨어산업협회, "2004 소프트웨어 사업대가기준 해설," 2004.3.

[5] 권기태 외, "소프트웨어 개발비 대가기준 개선 연구," 한국전산원 최종보고서, 2004. 11.

[6] Liming Zhu et al., "Tradeoff and Sensitivity Analysis in Software Architecture Evaluation Using Analytic Hierarchy Process," Software Quality Journal, 13, pp.357-375, 2005.

[7] T. Capers Jones, "Estimating Software Costs," McGraw-Hill, 1998.

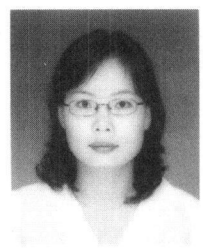
[8] Barry W. Boehm et al., "Software Cost Estimation with COCOMO II," Prentice-Hall, 2002.

[9] 조근태, 조용곤, 강현수, "계층분석적 의사결정," 동현출판사, 2003. 9.

[10] 김우제, 박찬규, 신수정, "AHP를 이용한 소프트웨어 개발비 보정계수 산정," IE Interface, Vol.17, Special Edition, pp. 1-10, 2004. 12.

[11] 권기태 외, "민간부문 정보화사업 비용자료 구축," 한국전산원 최종보고서, 2003. 11.

[12] Saltelli et al., "Sensitivity analysis," John Wiley and Sons, 2000. 8.



변 분 희

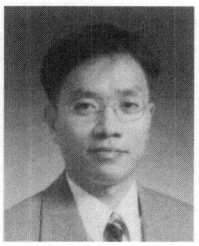
e-mail : bbh@kangnung.ac.kr

1996년 강릉대학교 컴퓨터공학과(학사)

2000년 강릉대학교 컴퓨터교육(석사)

2005년 강릉대학교 컴퓨터공학과 박사수료

관심분야 : 소프트웨어 비용산정, 소프트웨어 아키텍처



권 기 태

e-mail : ktkwon@kangnung.ac.kr

1986년 서울대학교 계산통계학과(학사)

1988년 서울대학교 계산통계학과(석사)

1993년 서울대학교 계산통계학과(박사)

1996년 Univ. of Southern California, Post-Doc.

1990년 9월 ~ 현 재 강릉대학교 컴퓨터공학과 교수

관심분야 : 소프트웨어 비용산정, 소프트웨어 메트릭스, 소프트웨어 아키텍처