이학박사 학위논문

COSMIC-FFP 기반 소프트웨어 개발노력 추정에 관한 연구

-가중치 적용을 통한 개선된 소프트웨어 개발노력 추정 모델을 중심으로-

2011년 2월

부 경 대 학 교 대 학 원

정보시스템협동과정

박 상 기

이학박사 학위논문

COSMIC-FFP 기반 소프트웨어 개발노력 추정에 관한 연구

-가중치 적용을 통한 개선된 소프트웨어 개발노력 추정 모델을 중심으로-

지도교수 박 만 곤

이 논문을 이학박사 학위논문으로 제출함.

2011년 2월

부 경 대 학 교 대 학 원

정보시스템협동과정

박 상 기

박상기의 이학박사 학위논문을 인준함

2011년 2월 23일

주 심 이학박사 이 경 현 ㊞

위 원 공학박사 김 창 수 ㊞

위 원 공학박사 정 민 수 ㊞

위 원 이학박사 정 혜 정 ㊞

위 원 이학박사 박 만 곤 ㊞

< 차 례 >

<표 차례>

<그림 차례>

A study on Software Development Effort Prediction Based on COSMIC-FFP

: Focus on improved software development effort estimation model using weight

Sang-Ki Park

Interdisciplinary Program of Information System,
The Graduate School, Pukyong National University

Abstract

Successful project is to develop the system which user is satisfied to by using of restricted manpower, period and cost. We chose a software size as a unit to estimate correct development effort, development costs and development period estimation.

It is possible for the accurate estimation of software size to allocate correct resource division and to definitely influence the software quality.

Consequently, The accurate estimation for development effort, cost and period is the essential element to accomplish the project successfully.

It is rather efficient that the estimation of software size is carried out at the early stage of the development. Line of code has been chiefly used to estimate the software size in korea because developers are easy to understand LOC and its meaning is clear. There are shortcomings in estimating LOC. First, It is hard to estimate LOC In the early stage of the software development. Second, LOC is influenced by the development environment.

The recent trend of software size estimation method prefers FPA to LOC. There are the problems of FPA. First, there exist measurement error between the actual value and the estimates. Second, it is impossible to apply for real-time system and embedded system.

However, The recent information system is getting larger in the portion of an embedded system or real-time system gradually.

Therefore, COSMIC is the established to cover for these disadvantages of function point in 1998 and COSMIC-FFP is developed for estimating real-time system and embedded system. There have been many researches on the software size estimation methods based on COSMIC-FFP, but there are a few models for estimation the development effort and it is necessary to develop the model.

The COSMIC-FFP technique has some advantages in appling several areas such as data management, real-time system, algorithmic software and others, but on the other hand, has some disadvantages in imposing the weight on function elements needed for estimating sizes. It is necessary for new method added complexity to COSMIC-FFP to quantify correctly the software size.

Consequently, First, we measure FP and COSMIC-FFP through the case study and propose the transfer model with the above two measurement results in this paper. Second, we propose estimation model for development efforts based on COSMIC-FFP by using of data from the case study and the weight-added model based on complexity to raise the correctness of the proposed model. We estimate the COSMIC-FFP according to the proposed model and evaluate the estimation model of development effort based on the weight-added COSMIC-FFP. Since the proposed models in this paper are based on the results from carrying out the actual project, it is possible to estimate the development period, costs and efforts correctly.

In the future, we will develop the general-use model through collecting various kind of an experiment data.

1. 서 론

1.1 연구 배경

성공적인 프로젝트란 제한된 인력, 기간 및 비용을 이용하여 사용자가 만족하는 시스템을 개발하는 것을 의미한다. 따라서, 정확한 개발노력, 개발비용, 개발기간의 예측이 필요하며, 이를 위해 소프트웨어 규모를 기본 단위로 채택하였다. 정확한 소프트웨어 규모의 예측은 향후 개발될 소프트웨어에 대한 정확한 자원 배분을 가능하게 하며 이는 소프트웨어 품질에 결정적인 영향을 준다. 따라서 소프트웨어 개발에 소요되는 노력과 비용, 기간 등의 예측은 해당 프로젝트를 성공적으로 수행하기 위한 필수적인 요소라고 할 수 있으며, 이러한 예측 활동은 프로젝트 초반에 수행되어야 보다 효과적일 수 있다. 이러한 이유로 소프트웨어 개발 초기단계에서 개발할 소프트웨어 규모를 정확히 예측하는 활동이 점차로 중요시 되고 있으며 [1], 계획단계에서 사용자가 제기한 요구사항 명세서(Requirement Specification)로부터 측정된 소프트웨어 규모를 이용해 개발에 소요될 노력(Effort)과 개발기간(Duration)을 추정함과 더불어 개발소요 비용(Cost)도 추정 가능하다.

우리나라에서는 소프트웨어의 규모를 산정하기 위해 주로 프로그램 라인 수(LOC, Line Of Code) 측정 방식을 사용해 왔는데, 이는 프로그램 소스 코드의 라인 수를 통해 소프트웨어의 규모를 산정하는 방법으로 개발자가 이해하기 쉽고 의미가 명확하기 때문에 지금까지 널리 사용되어 왔다. 하

지만 프로그램 라인 수 측정 방식은 소프트웨어 개발초기에 라인 수를 예측하기 어렵고 코딩이 완료된 시점에서 정확한 측정이 가능한 단점 및 소프트웨어 개발 언어나 환경에 영향을 받으므로 규모예측 방법의 근거가 희박하며, 추산시점에 하자가 발생할 수밖에 없는 등의 여러 가지 문제점을 가지고 있다[2]. 따라서, 소프트웨어 규모 산정 방식의 최근 동향은 프로그램 라인 수와 같은 개발자 관점의 접근방식 보다는 사용자가치 중심의 산정방식인 기능점수 분석기법(FPA, Function Point Analysis)를 보다 선호하고 있다. 기능점수기법은 사용자 관점에 기반을 두고 기능성으로 소프트웨어 규모를 정량화하는 방법으로 IFPUG(International Function Point User's Group)방법으로 현재 널리 알려져 있고 산업계에서 적용되고 있다. 또한, 소프트웨어 개발 초기단계에 적은 노력으로 소프트웨어 규모를 예측할 수 있다는 장점은 있으나 지나치게 단순화시킨 기능점수 예측 방식으로 인해 실제 산출결과와 예측된 기능 점수 사이에 측정 오차가 발생되며, 사용자 관점에서 기능으로 소프트웨어 규모를 정량화하는 방법으로 경영정보시스템(MIS: Managerment Information System) 소프트웨어에 기반을 두고 개발되어 실시간이나 임베디드, 공학계산 소프트웨어에는 적용이 불가능하다. 하지만, 최근 정보시스템은 임베디드 시스템이나 실시간 시스템의 비중이 점차적으로 확대되어가고 있는 추세이다. 따라서 이러한 기능점수의 문제점을 보완하기 위해 1998년 COSMIC(Common Software Management International Consortium)이 설립되어 경영정보시스템의 데이터 관리 위주인 기능점수기법을 제어관리 위주인 실시간 시스템과 내장형 소프트웨어로 적용범위를 확장하였다. 이를 완전기능점수 또는 COSMIC-FFP(Full Function Point)라 부른다[3].

측정된 소프트웨어 규모인 기능점수 FP(Function Point)를 개발하기 위해 투입될 노력을 추정하는 연구로는 Matson et al.[4], Albrecht[5][6],

Albrecht et al.[7], Kemerer[8][9], Low et al.[10] 등이 있다.

그리고, COSMIC-FFP기반의 소프트웨어 규모 측정 방법에 관한 많은 연구가 되고 있으나, COSMIC-FFP로 측정된 소프트웨어 규모에 기반하여 개발노력을 추정할 수 있는 모델 연구는 극소수에 불과함으로 모델 연구가 필요한 실정이다. 또한 완전기능점수는 복잡한 수학계산으로 구성된 알고리즘 위주의 소프트웨어 기능성을 측정하도록 설계되지 못한 단점을 갖고 있다. COSMIC-FFP기법은 소프트웨어 기능 프로세스의 컴포넌트 관점을 기술하고 있으나 규모 추정과 관련된 기능 요소들에 대하여 가중치를 부여하지 않아 모든 속성들을 개발하는데 동일한 노력이 투입됨을 의미하는 것으로써 현실적으로 적합하지 않은 문제점을 갖고 있다[11]. 소프트웨어의 규모를 보다 정확히 정량화하기 위해서는 기존의 COSMIC-FFP방법에 복잡도를 고려한 새로운 방법으로의 전환이 요구되고 있다.

1.2 연구 목적 및 기대효과

소프트웨어 규모에 대한 정량화는 소프트웨어 프로젝트에 소요되는 노력, 비용과 개발 일정을 적절히 추정하기 위한 핵심적인 인자 중 하나로 일반적으로 인식되고 있다[12]. 라인 수는 길이에 기반을 두고 있어 사용되는 프로그램밍 언어에 따라 다른 크기로 정량화되며, 코딩이 완료된 이후에야 정확한 크기를 측정 할 수 있는 취약점을 갖고 있다[13].

Albrecht[14]는 사용자에게 제공되는 소프트웨어의 기능성과 복잡도에 기반을 두고 규모를 정량화하는 기능점수 방법을 처음으로 제안하였다. 이 방

법은 데이터 관리 위주인 경영정보시스템 소프트웨어에 적합하도록 설계되어, 1990년대 들어 다른 소프트웨어 분야에서는 적용하기가 어렵다는 취약점을 나타내었다[15][16].

최근 들어 COSMIC-FFP가 제안되어 국제 표준으로 등재되었다. 그러나 이 방법으로 소프트웨어의 노력을 추정하는 모델에 관한 연구는 거의 미미한 실정이다. 또한 완전기능점수는 기능요소별 점수를 계산하는데 이용되는 속성들에 가중치가 적용되지 않아 모든 속성들을 개발하는데 동일한 노력이 투입됨을 의미하는 것으로써 현실적으로 적합하지 않은 문제점이 있다.

소프트웨어의 규모를 추정하면 소프트웨어 개발에 투입될 노력의 양, 개발비용 등을 추정할 수 있고 이를 계약단계부터 납품단계까지 관리적인 측면에서 활용될 수 있다. 개발노력을 추정하기 위해서는 먼저 소프트웨어의 규모를 추정해야 하며, 소프트웨어 규모는 길이, 기능, 복잡도로 표현될 수 있다. 이들 중 기능성 기반 소프트웨어 규모 측정 방법들은 복잡도와 기능을 함께 고려하고 있다.

따라서, 본 논문은 기능점수기법과 완전기능점수 기법에 따라 사례연구를 통해 개선된 방법으로 경험정보를 이용하여 소프트웨어 규모를 측정한다. 그리고, 측정된 두 종류의 소프트웨어 규모에 대해 회귀분석을 통하여 두 기법간의 변환 모델을 제시함으로써 경영정보시스템 뿐만아니라, 실시간 시스템 및 내장형 소프트웨어까지 소프트웨어 사업대가의 기준에 따라 소프트웨어 개발비를 산정할 수 있게 하였다.

본 연구를 통해 COSMIC-FFP방법에 복잡도를 고려한 새로운 방법을 제시함으로써 기존의 완전기능점수 규모 산정 방법에서의 문제점을 보완하여 소프트웨어 규모를 보다 정확하게 추정할 수 있으며, 이를 통해 프로젝트 개발 초기단계에서 소프트웨어 개발에 투입될 노력, 기간, 비용 등의 보다 정확한 예측이 가능하다. 본 논문에서 제안된 모델은 실제 프로젝트 수행

을 통해 산출된 데이터를 기반으로 하고 있어 프로젝트 현장에 적용이 가능하다.

1.3 논문 구성

본 논문에서는 첫째로 기존의 소프트웨어 규모 측정 방법 중 대표적인 기법인 기능점수 및 완전기능점수에 대한 추정기법을 고찰한다. 그리고 기능점수와 완전기능점수 사이의 차이점에 대해서 알아본다.

두 번째로, 사례연구를 통해 직접 기능점수와 완전기능점수를 측정해 봄으로써, 두 기능점수의 산출과정을 살펴보며, 산출된 두 기능점수를 이용하여, 기능점수와 완전기능점수 사이의 변환 모델을 제안한다.

세 번째로, 사례연구를 통해 산출된 데이터를 이용하여 COSMIC-FFP를 기반으로 한 개발노력 추정 모델을 제안하다. 또한 제안된 모델의 정확도를 제고하기 위하여 시스템 복잡도를 가중치로 부여하는 가중치 산정 모델을 제안하다. 그리고 제안된 모델에 따라 완전기능점수를 추정하고 가중치를 부여해 봄으로써 완전기능점수를 기반으로 한 개발노력 추정 모델에 대해 평가해 본다.

본 논문의 구성은 전체 6장으로 구성되어 있다. 제2장에서는 기능 기반 소프트웨어 규모 측정 기법에 관한 이론적 고찰로써 기존의 소프트웨어 규모 측정방법, 기능점수, 완전기능점수, 사용사례점수 및 기능기반 소프트웨어 규모 측정 방법들의 장·단점에 대해 간략하게 소개한다. 제3장에서는 실무사례연구를 통해 직접 기능점수와 완전기능점수를 측정하고 측정된 데

이터를 제시한다. 제 4장에서는 기능점수와 완전기능점수 사이의 변환과 관련된 연구를 살펴보며, 실무사례연구를 통해 측정된 데이터를 이용하여 두 가지 기법 사이의 변환 모델을 제시한다. 제5장은 본 논문에서 제안하는 COSMIC-FFP를 기반으로 한 개발노력 추정 모델에 대해 설명한다. 그리고 COSMIC-FFP의 문제점에 대해 기술하고, COSMIC-FFP가 갖고 있는 문제점을 해결하여 보다 현실적으로 타당하게 적용할 수 있는 시스템 복잡도를 고려한 소프트웨어 개발노력 추정 모델을 제안하며, COSMIC-FFP 기반 개발노력 추정 모델에 가중치를 적용할 수 있도록 시스템 복잡도 추정 모델도 제안한다. 끝으로 6장에서는 결론 및 향후 연구 방향에 대해 제시한다.

2. 기능 기반 소프트웨어 규모 측정 기법에 관한 이론적 고찰

소프트웨어 측정분야는 프로세스와 제품 메트릭스로 구분된다. 제품 메트릭스는 다시 내부 속성과 외부 속성으로 구분할 수 있다. 소프트웨어의 내부 속성으로는 규모가 있다. 규모는 길이, 기능 복잡도, 재사용과 같은 다른 속성들로 표현될 수 있다. 규모에 대한 단순한 측정은 노력, 생산성, 비용과 품질에 대한 충분한 정보를 제공하지는 못한다. 길이는 제품의 물리적인 측면의 규모이며, 기능점수는 기능 측면의 규모이다.

현재까지 소프트웨어 규모를 측정하는 방법으로는 길이에 기반을 둔 LOC, 기능에 기반을 둔 기능점수, 객체점수, 특징점수 등의 방법들이 소개되어 왔다. 본장에서는 이러한 대표적인 소프트웨어 규모 측정 방법들의 특징 및 장·단점에 대해 살펴보고, 본 논문에서 제안된 모델의 기반이 되는 완전기능점수의 특성에 대해 설명한다.

2.1 기존의 소프트웨어 규모 측정 기법

최근 들어, 소프트웨어 개발기간과 비용 추정 결과는 크게 부정확하고, 불충분한 품질을 갖고 있으며, 또는 대형 소프트웨어 프로젝트의 1%만이 계획된 기관과 예상 비용한도 내에서 고객의 요구사항을 만족 시키고 제한된 기간 내에 프로젝트를 완료 하였으며, 대부분의 프로젝트들은 1년 이상의 일정이 지연되거나 초기 예상 비용의 2배 정도가 초과 되었다[17]. 이 같은 이유로 소프트웨어 개발에 소요되는 비용, 인력과 개발기간을 추정하기 위해서는 소프트웨어 규모를 측정해야만 한다. 따라서 본 장은 소프

트웨어 규모가 가지는 의미와 소프트웨어 규모 측정 방법들에 대해서 알아본다.

소프트웨어 규모는 프로젝트 수행에 필요한 자원의 유형과 정도 뿐 아니라 투입노력과 소요기간을 계산하기 위해 필요하며, 이에 대한 산정은 프로젝트 비용을 결정하는데 결정적 요인으로 작용한다[18]. 소프트웨어 규모 측정의 척도를 간략히 살펴보면 다음과 같은 기법이 있다.

먼저, 소프트웨어 규모 측정에 사용되어지는 기법 중에서 가장 폭 넓게 사용되는 척도가 기능점수와 라인 수이다. 라인 수 기반의 COCOMO I은 프로젝트에 소요되는 노력의 10%에서 15%만이 소요되는 코딩단계에만 한정되어 초점을 맞춘 기법으로 대형 프로젝트에 대해 평균적으로 초기 추정은 매우 부정확하다. 이 기법은 개발자 관점에서 본 것이며, 사후 추정으로 기법으로 개발 생명주기의 후반부에 가서야 비로써 획득이 가능하다. 라인수는 다음과 같은 특징을 가진다.

첫째, 프로그래밍 형태에 매우 의존적이다. 똑같은 기능을 가지는 프로그램에 대해 프로그래머가 코딩을 어떤 형태로 하는가에 따라 라인수가 상당히 다르게 측정 될 수 있다. 따라서, 측정치에 대한 객관성이 떨어진다. Jones[19]는 라인을 계산하는 방법에 대해 11개의 변형된 방법을 확인했으며, 라인 계산 방법에 따라 약 500%의 불확실성을 가지고 있음을 지적 했다[20].

둘째, 기술에 의존적이기 때문에 같은 기능에 대하여 종류가 다른 언어로 개발되어진 각각의 프로젝트에 대해서 서로 비교가 어렵다. 만약, C++과 Java처럼 두 언어는 표면상 매우 밀접한 관계를 가지고 있지만 각각의 언어로 표현된 코딩 라인수 비교를 통해 소프트웨어 규모를 측정하기는 힘들다. 그리고 3세대 언어는 라인당 투입되는 시간이 4세대 언어보다 많으며, 동일한 기능을 제공할 경우 4세대 언어가 보다 적은 라인수를 가짐을 확실

히 예상 할 수 있다.

셋째, 하나의 프로젝트가 2가지 이상의 언어가 혼합되어 개발되어졌을 때, 다른 프로젝트와 비교하기는 거의 불가능하다. 한 종류의 언어로 개발되어진 소프트웨어도 프로그래밍 형태나 언어 종류가 다를 경우 비교의 객관성이 떨어진다. 이에 비해 소프트웨어가 2종류 이상의 언어로 개발 되었다면, 비교 경우의 수가 2^n(n= 개발 언어 수)이 될 것이다.

넷째, 라인수는 개발이 완료된 이후 코드를 계산함으로 계획이나 견적 단계에서는 사용되어질 수 없다. 하지만 소프트웨어 개발노력은 프로젝트 착수 초기에 보다 정확히, 추정하여 사업관리나 자원 재분배 측면에서 활용할 수 있어야 한다.

다섯째, 개발자의 관점에서 소프트웨어의 규모를 측정한다. 위에서 보듯이 라인수라는 것이 개발자가 프로그램을 개발하는데 어느 정도의 노력이 투입되는가를 판단하는 기준으로 작용하는 것이다. 이는 사용자 입장을 전혀 반영해 주지 못한다. 또한, 특정한 하나의 관점인 코드의 길이만을 고려하고, 소프트웨어의 기능성 또는 복잡도를 고려하지 못해 개발 소프트웨어의 특성을 정확히 반영하지 못하는 단점을 가지고 있다.

라인 수를 이용하여 소프트웨어 개발 노력 E를 추정하는 모델은 식(1) 또는 (2)의 형태를 취한다.

$$E = a + b \times KLOC^{C} \qquad (1)$$

$$E = a \cdot KLOC^{b} \qquad (2)$$

식에서 E는 예측된 개발노력으로 Man-Months로 측정되고, a, b, c는 상수이며, KLOC는 최종 코딩된 라인수의 추정된 KLOC(Thousands of Line

Of Code)의 수이다.

예로 Bailey-Basili[21]은 $E=5.5+0.73(KLOC)^{1.16}$의 모델을 제안하였다. Walston-Felix[22]은 IBM의 Federal System Division에서 개발된 60개의 소프트웨어 프로젝트들을 이용해 소프트웨어 개발노력을 추정하는 모델로 개발노력의 주요 인자로서의 LOC를 고려하여 $E=a\cdot KLOC^{b}$ 형태의 모델로 모수를 추정하였으며 그 결과 $E=5.2(KLOC)^{0.91}$의 모델을 얻었다. Boehm은 제품 개발에 있어서 프로젝트의 복잡도를 결정하는데 도움을 주는 Organic, Semi-detached와 Embedded 모드로 분류하여 개발 노력을 추정하는 $E=aKLOC^{b}$의 모델을 제안하였다. 여기서 a, b는 각 모드와 모델 수준에 대해 결정되는 상수이다. 3가지 모드에 대해 Orgnic(Simple)은 $E=3.2(KLOC)^{1.05}$, Semi-detached(Average)은 $E=3.0(KLOC)^{1.12}$, Embedded(Complex) 모델은 $E=2.8(KLOC)^{1.20}$을 제안하였다.

반면에, 기능점수는 다음과 같은 특징을 가진다. 첫째, 기능점수는 소프트웨어에 의해 제공되는 기능의 규모를 측정하며, 기능상으로는 데이터와 이러한 데이터로 수행되는 처리기능으로 측정된다. 측정되는 도구와 사용되는 기술에 독립적이므로 다양한 조직과 프로젝트사이의 비교를 위한 일관성을 제공해 줄 수 있다.

둘째, 기능점수는 요구분석 단계에서 사용자 요구사항 명세를 기반으로 측정되므로, 계획과 견적을 위해서 사용될 수 있다.

셋째, 사용자의 관점에서 기능을 중심으로 소프트웨어의 규모를 측정한다. 여기서 사용자 관점은 프로그램 사용자에 의해 업무기능이 정의되고, 검증되어 짐을 의미한다. 이것은 사용자의 용어를 사용하여 사용자 업무 요구를 체계적으로 기술한 것을 말한다. 물리적 형태에서 매우 다양하게 나타난다. 예를 들면, 처리 일람표, 제안서, 요구문서, 외부명세, 상세명세서, 사용자 핸드북 등이다. 개발자는 사용자의 요구를 해결하기 위해 사용자 정

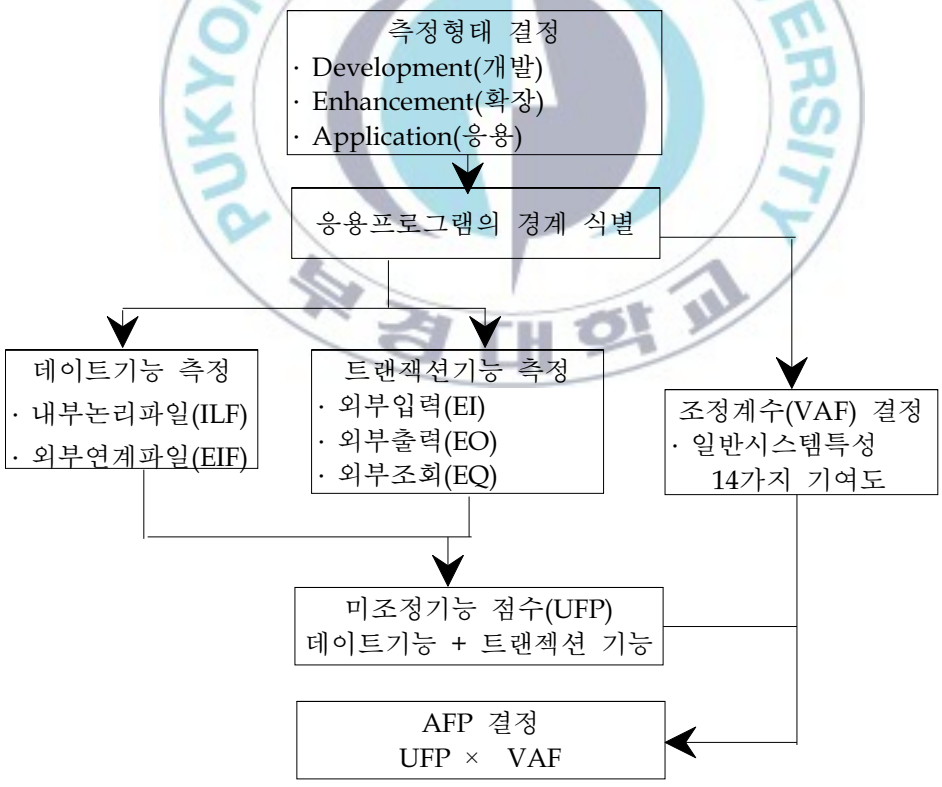
보를 정보 기술 용어로 전환한다. 기능점수는 언어라는 정보를 사용하여 사용자와 개발자의 공통 인식에 기초하여 계산된다. 따라서, 위와 같은 기능점수의 장점 때문에 최근에는 기능점수를 기반으로 한 소프트웨어 규모 측정 기법이 널리 확산되어 사용하고 있으며, 지경부에서 제시한 소프트웨어사업대가의 기준에서도 이를 채택하고 있다. 그리고, 기능점수의 단점을 극복하기 위해 특징점수, MKII 기능점수, 3차원 기능점수와 완전기능점수 등이 제안되었다. 특징점수는 관리정보, 실시간, 내장형, 통신, 공정관리 시스템, CAD, 인공지능, 이산형 시뮬레이션, 수학과 공학 등 모든 분야의 응용 프로그램에 적합하도록 제안되었으며, 기능점수 기법의 상위집합이다. 이 방법은 기능점수의 5개 기능 요소에 알고리즘을 추가로 고려하였으며, 기능점수와 차별화를 위해 특징점수라고 칭하였다. MKII 기능점수는 MIS 소프트웨어의 내부처리 복잡도를 보다 좋게 고려하기 위해 제안되었다. 또한, 80년대 후반과 90년대 초반에 유행한 구조적 분석방법에 일치하도록 Albrecht의 기본적인 개념을 갱신하였다. 이 방법은 MIS 소프트웨어가 입력, 처리와 출력 컴포넌트를 가진 논리적 트랜잭션으로 구성되어 있다고 간주하였다. 논리적 트랜잭션의 처리를 참조하는 엔티티 타입들의 수를 계산하여 처리 컴포넌트 규모를 결정하였다. Albrecht의 기능점수 방법은 소프트웨어의 일부분에 대해 논리적 파일을 한번 계산하는데 비해 MKII 기능점수 방법은 각 논리적 트랜잭션을 참조할 때마다 엔티티 타입을 매번 계산한다. 또한 기능점수의 14개 일반적인 시스템 속성에 6개의 복잡도 요소를 추가 하였다. 그러나 이것은 더 이상의 의미가 없어 몇 년 후 삭제되었다. 3D 기능점수는 Albrecht의 기능점수 방법을 실시간 시스템으로 확장하였으며, 모든 문제영역에 적합하도록 기술에 의존하지 않는 규모 추정 필요성에 따라 제안되었다. Albrecht의 기능점수는 응용 프로그램이 기능과 데이터 컴포넌트로 표현될 수 있다고 본 반면 3D 기능점수는 데이터,

기능과 제어의 3가지 영역으로 표현될 수 있다는 논리에 기초하고 있다. 데이터가 우세한 문제영역은 정보 시스템과 비즈니스 소프트웨어 환경으로 기능점수 5가지 기능요소를 적용한다. 기능이 우세한 문제영역은 과학, 공학 환경으로 입력 데이터를 출력 데이터로 변환하는데 요구되는 내부처리 표현 기능의 수와 복잡도, 처리를 관장하는 의미 문장의 수로 표현된다. 제어가 우세한 문제영역은 실시간 환경으로 시스템 상태가 전이로 표현된다. 객체점수는 3세대 언어에 적합하도록 화면, 보고서와 3세대 컴포넌트에 대한 복잡도를 고려하여 규모를 측정하였다[23].

2.2 기능점수(FP, FUNCTION POINT)

기능점수[24]는 70년대 중반 Allen Arbrecht에 의해 Joint/Share 학술발표에서 처음으로 소개되었다. 1983년에 IBM GUIDE 협회 안에 설립된 워킹 그룹을 중심으로 정비와 활동이 시작되어, 1986년에 IBM GUIDE 협회로부터 독립하여 세계적인 FPA 이용자 모임 IFPUG가 결성되었다[1]. IFPUG에서 공식적으로 측정 규칙에 대한 정련작업과 정의작업을 수행하여, IFPUG법을 만들었으며, 이는 현재 우리가 가장 많이 사용하고 있는 FPA 방법이다[25].

이 절에서는 소프트웨어 규모 측정을 위해 IFPUG 측정 수행 지침 4.1에 따라 [그림 1]과 같은 순서로 기능점수 산정 절차를 설명한다[24].



[그림 1] 기능점수 산정절차

단계 1. 기능점수 측정형태는 개발프로젝트 기능점수(DFP, Developemnt Function Point), 확장프로젝트 기능점수(EFP, Enhancement Function Point), 응용 기능점수(AFP, Application Function Point)로 나누어지며, 이 3가지 유형 중에서 프로젝트 목적에 부합되는 형태를 선택한다.

단계 2. 응용프로그램 경계를 식별하며, 응용프로그램 경계는 측정 대상 소프트웨어와 사용자 간 경계로, 내부 응용프로그램과 외부 사용자 세계 간 개념적 인터페이스를 나타내며, 상세한 사용자 요구사항 정의에 의하여 응용프로그램 경계를 식별한다. 어플리케이션 경계는 사용자 관점에 기초해 결정되는 것으로, 그 초점은 사용자가 무엇을 이해하고 기술하느냐에 있다.

단계 3. 데이터기능 측정은 최종 기능점수 산정을 위한 응용프로그램에서 사용되고, 조정된 데이터의 기여를 말하며, 내부논리파일(ILF, Internal Logical Files)과 외부연계파일(EIF, External Interface File)로 분류된다.

ILF와 EIF는 각각에 관련된 데이터항목유형(DET, Data Element Type)과 레코드항목유형(RET, Record Element Type)의 숫자에 기초하여 기능 복잡도가 결정되어진다. ILF와 EIF의 RET와 DET의 수를 파악하여 [표 1]에 따라 복잡도를 결정한다.

[표 1] ILF, EIF 기능 복잡도

레코드요소 유형의 개수	데이터요소유형의 개수		
	1~19	20~50	51이상
1	낮음	낮음	보통
2 ~ 5	낮음	보통	높음
6이상	보통	높음	높음

[표 2]는 기능 복잡도에 ILF, EIF 각각의 가중치를 제시하였다.

[표 2] 가중치 W_{ij}

기능요소형태	기능 복잡도		
	낮음	보통	높음
ILF	× 7	× 10	× 15
EIF	× 5	× 7	× 10
EI	× 3	× 4	× 6
EO	× 4	× 5	× 7
EQ	× 3	× 4	× 6

단계 4. 트랜잭션기능을 측정한다. 트랜잭션기능은 처리 데이터에 대해 사용자에게 제공되는 기능성을 의미한다. 트랜잭션기능은 외부입력(EI, External Inputs), 외부출력(EO, External Output), 외부조회(EQ, External Inquiries)로 구성된다. EI, EO, EQ 각각의 FTR과 DET의 수를 파악하여 [표 3]과 [표 4]에 따라 복잡도를 결정한다. 그리고, [표 2]에 의해 EI, EO, EQ 각각의 복잡도에 따른 가중치를 결정한다.

[표 3] EI 복잡도

참조파일 유형의 개수	데이터요소유형의 개수		
	1~4	5~15	16이상
0 ~ 1	낮음	낮음	보통
2	낮음	보통	높음
3이상	보통	높음	높음

[표 4] EO, EQ 복잡도

참조파일 유형의 개수	데이터요소유형의 개수		
	1~5	6~19	20이상
0 ~ 1	낮음	낮음	보통
2 ~3	낮음	보통	높음
4이상	보통	높음	높음

단계 5. 데이터 기능과 트랜잭션 기능 가중치를 합하여 다음과 같은 방법으로 기능점수를 계산한다. 기능요소(ILF, EIF, EI, EO, EQ)들을 i, 복잡도 정도를 j라고 하면, [표 1], [표 3], [표 4]에 의해 복잡도 정도 j이고 i번째인 기능요소를 계산 한 값을 Z_{ij}라하고 각 복잡도 정도에 대한 [표 2]의 가중치 W_{ij}와 선형결합 시키면 식(3)의 UFP (Unadjusted FP)가 계산된다[26].

$$UFP = \sum_{i=1}^{5}\sum_{j=1}^{3} W_{ij}Z_{ij} \qquad (3)$$

단계 6. 측정되는 응용프로그램의 일반적 기능에 등급을 부여한 14개의 일반 시스템 특성(GSC, General System Characteristics)에 기반을 두어 보정계수(VAF, Value Adjustment Factor)를 계산한다. VAF는 다음과 같이 응용프로그램의 일반적인 기능성 평가 14개의 GSC로 구성된다.

① 데이터 통신 (Data Communication) : 어플리케이션이 직접적으로 프로세서와 통신하는 정도를 기술한다. 어플리케이션에서 사용되는 데이터와 정보통제는 통신장비를 통하여 송・수신 된다.

② 분산 데이터 처리 (Distributed Data Processing) : 어플리케이션이 어플리케이션 컴포넌트들 사이에서 데이터를 전송하는 정도를 말한다.

③ 시스템 성능 (Performance) : 어플리케이션 개발에 영향이 있는 시스템 응답시간과 처리 성능에 대한 정도를 말한다.

④ 가중된 사용 형상 (Heavily Used Configuration) : 어플리케이션 개발에 영향을 주는 컴퓨터 자원 제약의 정도를 말한다.

⑤ 트랜잭션 비율 (Transaction Rate) : 어플리케이션 개발에 영향을 미치는 업무 처리량에 대한 정도를 말한다.

⑥ 온라인 데이터 입력 (OnLine Data Entry) : 대화식 트랜잭션을 통해 입력되는 데이터 정도를 말한다.

⑦ 최종 사용자 효율성 (End-User Efficiency) : 최종 사용자 효율성은 인간적인 요소 및 사용자의 어플리케이션 사용의 편리성의 고려정도를 말한다.

⑧ 온라인 갱신 (OnLine Update) : 온라인 갱신은 내부논리 파일이 On-Line으로 갱신되는 정도를 말한다.

⑨ 처리 복잡도 (Complex Processing) : 처리 복잡도는 처리 로직이 어플리케이션 개발에 영향을 주는 정도를 말한다.

⑩ 재사용성 (Reusability) : 재사용성은 어플리케이션과 소스 코드를 다른 어플리케이션에서 사용 지원을 위해 특별하게 설계, 개발되어야 하는지에 대한 정도를 말한다.

⑪ 설치 용이성 (Installation Ease) : 설치용이성은 이전 환경으로부터 변환이 어플리케이션 개발에 영향을 미치는 정도를 말한다.

⑫ 운영 용이성 (Operational Ease) : 운영용이성은 어플리케이션의 개시, back-up, 복구 프로세스 등 운영성 측면의 고려사항에 대한 정도를 말한다.

⑬ 다중 설치성 (Multiple Install) : 다중설치성은 개발되는 어플리케이션이 복수조직 및 복수장소를 고려하는 정도를 말한다.

⑭ 변경 용이성 (Facilitate Change) : 변경용이성은 어플리케이션이 프로세스 로직이나 데이터 구조의 변경 용이성을 위해 개발되어야 하는 정도를 말한다.

위 14개의 GSC 각각에 대한 영향도(DI, Degree of Inference)의 범위는 '없음(0)'에서 '강함(5)'까지 6등급으로 구분되며, 14개의 일반 시스템 특성의 영향도를 합해 총 영향도(TDI, Total Degree of Inference)를 계산한다. 즉, $TDI = \sum_{i=1}^{14} DI_i$, TDI를 식(4)에 대입하여 VAF를 계산 할 수 있다.

$$VAF = (TDI \cdot 0.01) + 0.65 \qquad (4)$$

단계 7. 기능점수 분석의 마지막 단계로, 측정 타입 유형별(DFP, EFP, AFP)로 구성 요소 간 계산식에 의해 조정기능점수를 도출한다. 조정 기능점수(AFP, Adjusted FP)는 UFP에 조정계수인 VAF를 곱하여, 식(5)와 같이 계산 할 수 있다.

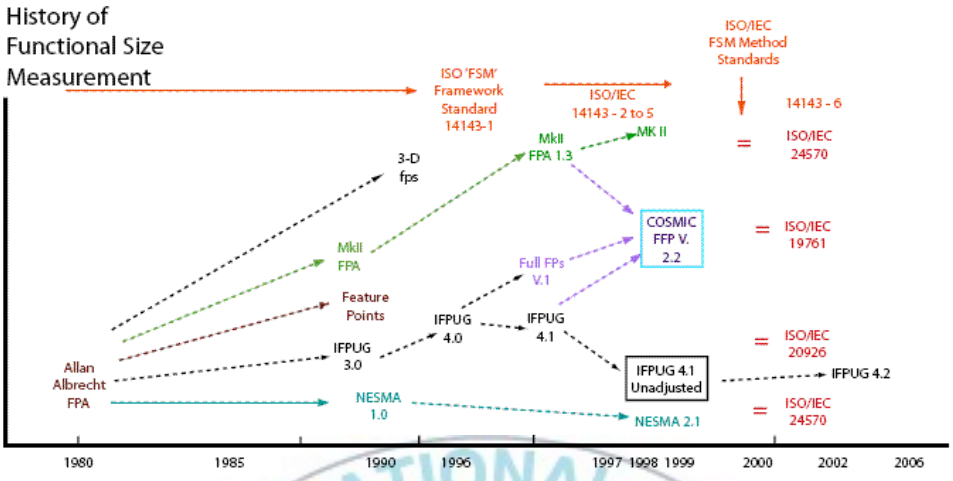
$$AFP = VAF \cdot UFP = VAF \cdot \sum_{i=1}^{5}\sum_{j=1}^{3} W_{ij}Z_{ij} \qquad (5)$$

2.3 완전기능점수와 사용사례점수

최근 들어 소프트웨어 규모를 추정하는 방법으로 완전기능점수방법과 사용사례점수 방법이 출현하였다. 소프트웨어 측정방법의 새로운 세대를 설계하고 판로를 개척하기 위해 산업계의 지원을 받아 COSMIC(Common Software Measurement International Consortium)이 1998년에 설립되었다. 그 결과 경영정보시스템 영역인 데이터 관리 위주에서 제어관리 위주인 실시간 시스템과 내장형 소프트웨어에도 적용할 수 있도록 적용 범위를 확장하였으며, 실용성이 입증되었다[27]. 이를 FFP 또는 COSMIC-FFP라 불리운다. 또한 객체지향 소프트웨어 개발 분야에서는 요구사항의 보다 정확한 도출을 위해 사용사례를 도입하고 있으며, 이에 기반하여 UML에 따른 분석 및 설계를 수행하고 있다.

가. 완전기능점수(COSMIC-FFP, COSMIC-Full Function Point)

1998년에 COSMIC이 설립되었고, 소프트웨어 기능 크기 측정 방법의 새로운 세대에 기초를 둘 수 있는 기본적인 원리를 제안하였으며, 1999년에 제안된 원리를 구현한 측정방법인 COSMIC-FFP V.2.2을 발표 하였다. 그리고, 소프트웨어의 크기를 정량화하는 것은 소프트웨어 프로젝트의 노력, 비용, 개발기간을 평가하는 키 중의 하나로 일반적으로 인정된다[28].
소프트웨어의 기능성에 기반하여 소프트웨어의 규모를 추정하기 위한 기법들의 변천과정을 [그림 2]에 기술하고 있다.



[그림 2] 기능적인 규모측정 방법의 변천 과정

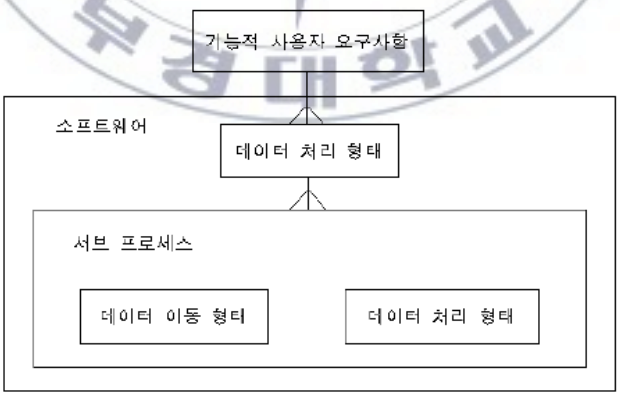
COSMIC-FFP에 기반한 소프트웨어 규모의 단위는 CFSU(Cosmic Functional Size Unit)로 표기한다. 그리고, COSMIC-FFP 기능 규모는 ISO/IEC 19761에 제안한 방법에 따라 측정 할 수 있다[28].

완전기능점수 방법은 다층 또는 다단계로 연결된 소프트웨어 구조의 임의의 계층에 있는 규모도 측정할 수 있으나 복잡한 수학계산으로 구성된 알고리즘 위주의 소프트웨어에 대한 기능성을 측정하도록 설계되지는 못하였다. 2000년에 다양한 조직, 다양한 영역에서 개발되는 실시간과 내장형 소프트웨어에 대해 적용 타당성이 입증되었고, 경영정보시스템 소프트웨어에 대해서도 유사한 결과를 얻었다. 또한 COSMIC-FFP는 소프트웨어 기능 규모 측정에 대한 ISO/IEC-14143 표준을 만족하도록 설계되었으며, 2003년 국제 표준화인 ISO/IEC-19761로 등재 되었고 ISBSG(International Software Benchmarking Standards Group)에 의해 인정되었다[29][30].

ISO/IEC-19761은 경영정보시스템의 소프트웨어를 정량화하는 기능점수 방법에 기반을 두고, 추가로 실시간 소프트웨어를 정량화 할 수 있도록 6개

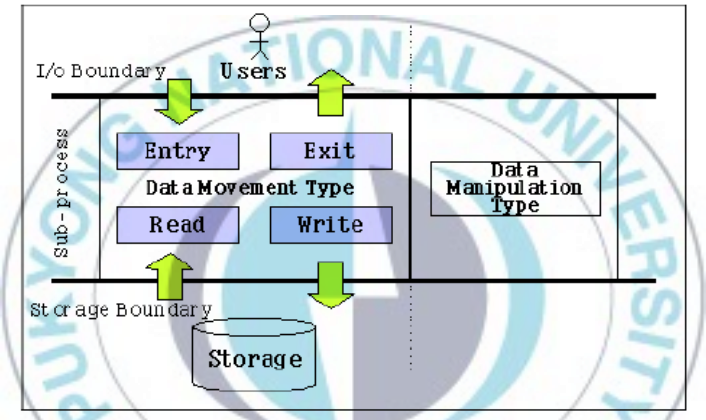
의 새로운 컴포넌트를 추가하여 몇몇 회사에서 적용성이 검증되었다. 완전기능점수 버전 2는 경영정보시스템과 실시간 소프트웨어에 동등하게 적용될 수 있는 새로운 방법이다. 이 방법은 기능 사용자 요구사항을 기능 프로세스들로 분할하고 다시 기능 서브프로세스들로 분할한다. 기능 서브프로세스는 외부와 관련된 입력과 출력, 파일과 관련된 읽기와 쓰기로 데이터 이동현태만을 고려하고 있으며, 각각은 관련된 데이터 조작인 알고리즘을 가지고 있다고 가정한다.

완전기능점수는 [그림 3]과 같이 기능 요구사항에 기반을 둔 일반적인 소프트웨어 모델이다. 이 모델에 의하면 소프트웨어 생명주기 초기 단계에서도 소프트웨어는 사용자의 기능 요구사항들로 취급하고 있으며, 이들은 기능 프로세스로 구현된다. 기능 프로세스는 다시 데이터 이동형(Data Movement Type)과 데이터 처리형(Data Manipulation Type)으로 구성된다.



[그림 3] COSMIC-FFP의 일반적인 모델

기능 규모를 측정하기 위해 [그림 4]와 같이 기능 프로세스는 하나의 자극을 주는 사건(Triggering Event)에 의해 활성화되기 때문에 데이터 처리형은 취급하지 않고 데이터 이동형만을 고려하였다. 자극이 주어지면 입력 데이터(입력, 읽기)를 받아들여 데이터를 처리하여 출력 데이터(쓰기, 출력)를 생성한다.

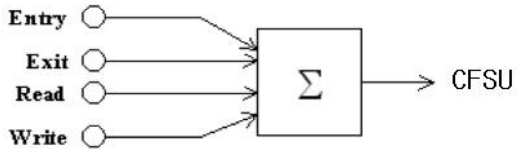


[그림 4] COSMIC-FFP의 서브 프로세스 형

[그림 4]와 같이 COSMIC-FFP는 데이터 이동형을 4개의 속성으로 분류한다[29]. 소프트웨어 측면에서 바라볼 때, 입구와 출구는 사용자와 데이터의 속성을 주고 받는 것이며, 읽기와 쓰기는 저장장치와 데이터 속성을 주고 받는 형태이다. COSMIC-FFP에 기반한 소프트웨어 규모의 단위는 CFSU로 표기한다. 신규로 개발되는 경우, 주어진 기능 프로세스의 기능 규모는 식(6)을 사용하여 CFSU로 계산되고 계산과정은 [그림 5]에 제시되어 있다.

$$SizeCFSU = \Sigma(Ne*Eus) + \Sigma(Nx*Xus) + \Sigma(Nr*Rus) + \Sigma(Nw*Wus) \quad (6)$$

여기서, Ne는 입력의 수, Eus는 입력 단위 크기(Unit Size), Nx는 출력의 수, Xus는 출력 단위 크기, Nr은 읽기의 수, Rus는 읽기 단위 크기, Nw는 쓰기의 수, Wus는 쓰기 단위 크기를 의미한다.



[그림 5] COSMIC-FFP계산 과정

기존에 개발된 기능 프로세스의 변경이 발생한 경우는 식(7)에 의해 기능 규모가 계산된다.

$$Size_{CFSU} = \sum Size(Added) + \sum Size(Changed) + \sum Size(deleted) \quad (7)$$

여기서, Size Added는 추가된 기능의 수, Size Changed는 변경된 기능의 수, Size Deleted는 삭제된 기능의 수를 의미한다.

개발이 완료된 소프트웨어를 운영하면서 발생하는 유지보수 활동은 정정 유지보수(Corrective Maintenance), 완전 유지보수(Perfective Maintenance)와 적응 유지보수(Adaptive Maintenance)로 구분된다. 정정 유지보수는 운영 단계에서 발견된 결함을 제거하는 활동이며, 새로운 기능을 추가하여 성능을 향상시키는 경우를 완전 유지보수라 한다. 이에 비해, 데이터베이스 또는 운영체제의 업그레이드, 컴파일러 버전 변경 등과 같은 운영환경 변화로 요구되는 소프트웨어 수정을 적응 유지보수라 한다. 운영단계에서 유지보수로 인해 기존 프로그램의 기능들이 추가(Added), 변경(Changed) 또는 삭제(Deleted) 되는가에 관심을 가진다. 따라서 식(7)은 유지보수 프로

젝트라 할 수 있으며, 이에 비해 식(6)은 순수한 개발 프로젝트로 생각 할 수 있다.

완전기능점수에 기반한 소프트웨어 규모는 CFSU로 표기하며, 임의의 i번째 계층에 대한 CFSU는 [그림 5]의 계산 과정에 따라 식(6)과 같이 계산된다[31][32].

나. 사용사례점수(UCP, Use Case Point)

사용사례는 사용자가 추구하는 목표(Goals) 또는 필요로 하는 니즈(Needs)를 충족하기 위해 시스템을 사용하는 이야기(Stories)로 요구사항을 이해하고 기술하는 탁월한 기법으로 알려져 있다[33]. 사용사례는 1992년 Jacobson[34]에 의해 제안되었으며, 객체지향 소프트웨어 개발 전 단계에 포함되어 있다[33][35].

사용사례 모델은 사용사례 설명서(Use Case Description)와 사용사례도(Use Case Diagram)로 구성되어 있으며, 일반적으로 불리우는 사용사례는 사용사례 설명서를 의미하며, 사용사례도는 사용사례 설명서를 보충 설명하기 위한 용도로 사용된다[33]. 주 행위자(Primary Actor)의 사용자 목표를 만족시키기 위해 사용사례가 정의되기 때문에 사용사례 모델링은 먼저 시스템의 범위(Boundary)를 선택하고, 주 행위자를 식별한 후, 주 행위자의 사용자 목표를 식별하여 사용자 목표를 만족시키는 사용사례를 정의(기술)하는 단계를 거친다.

사용사례 점수를 계산하는 방법은 [그림 6]에 제시하였으며, 다음 순서로 계산된다[31][33][35].

단계 1. [표 5]에 의해 액터 복잡도인 UAW를 계산한다.

[표 5] UAW 계산

Actor Type	Description	Weight Factor (①)	Number of Actors (②)	UAW (①×②)
Simple	응용 프로그램밍 인터페이스(API)를 통해 인터페이스된 타 시스템	1		
Average	대화식 또는 TCP/IP등의 프로토콜을 통해 상호작용하는 타 시스템	2		
Complex	GUI 또는 Web페이지를 통해 상호작용하는 사용자	3		
UAW(Unadjusted Actor Weights) = $\sum_{i=1}^{3} UAW_i$				

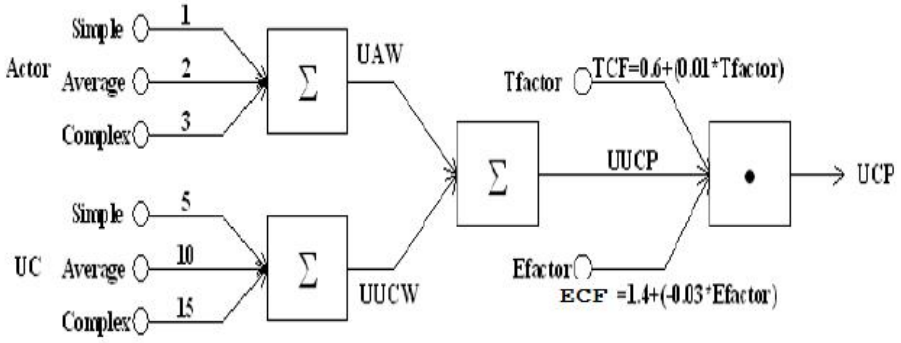
단계 2. [표 6]에 따라 사용사례 복잡도인 UUCW를 계산한다.

[표 6] UUCW 계산

Use Case 형태	방 법		Weight Factor (①)	Number of Actors (②)	$UUCW_i$ (①× ②)
	Number of Transaction (Key Scenario)	Number of Classes			
Simple	≤ 3	< 5	5		
Average	4 ~ 7	5 ~ 10	10		
Complex	≥ 8	> 10	15		
UUCW(Unadjusted Use Case Weights) = $\sum_{i=1}^{3} UUCW_i$					

단계 3. 식(8)과 같이 UUCP를 계산한다.

$$UUCP = UAW + UUCW \quad (8)$$



[그림 6] UUCP 계산과정

단계 4. [표 7]의 기술적 요인(TFactor; Technical Factor)을 계산 후 식(9)에 의거하여 생산성에 영향을 미치는 기술적 복잡도 요인(TCF : Technical Complexity Factor)을 계산한다.

$$TCF = 0.6 + (0.01 \cdot TFactor) \quad (9)$$

단계 5. [표 8]의 환경적 요인(Efactor : Environmental Factor)를 계산 후 식(10)에 의거 생산성에 영향을 미치는 환경적 복잡도 요인(ECF : Environmental Complexity Factor)을 계산한다.

$$ECF = 1.4 + (-0.03 \cdot EFactor) \quad (10)$$

단계 6. 식(11)은 조절된 사용사례 점수를 계산한다. AUCP를 UCP라 칭한다.

$$AUCP = UUCP \times TCF \times ECF \quad (11)$$

TFactor와 EFactor에서 적용되는 가중치 할당 값은 기능점수의 DI 계산에 적용된 기준과 동일하다.

[표 7] TFactor 계산

기술적 요인	Description	Weight Factor (①)	Number of Actors (②)	TF_i (①× ②)
T1	분산시스템	2	0 ~ 5	
T2	응답시간 또는 처리시간	1		
T3	사용자 능력	1		
T4	내부처리 복잡도	1		
T5	코드 재사용성	1		
T6	설치 용이성	0.5		
T7	사용 용이성	0.5		
T8	휴대용	2		
T9	변경 용이성	1		
T10	병렬처리	1		
T11	특수 보안	1		
T12	제3자 직접 접근성	1		
T13	특별한 사용자 교육시설	1		
TFactor(Total Technical Factor) = $\sum_{i=1}^{13} TF_i$				

[표 8] EFactor 계산

환경적 요인	Description	Weight Factor (①)	Number of Actors (②)	EF_i (①× ②)
E1	RUP 친숙도	1.5	0 ~ 5	
E2	해당분야 경험	0.5		
E3	객체지향 경험	1		
E4	분석책임자 능력	0.5		
E5	동기부여	1		
E6	요구사항 안정 정도	2		
E7	비상근 개발요원	-1		
E8	프로그램 언어 어려움	-1		
EFactor(Total Environmental Factor) = $\sum_{i=1}^{8} EF_i$				

사용사례는 시스템이 요구하는 특징의 관점이 아니라 사용자가 시스템을 실제로 어떻게 사용하는가의 관점에서 요구사항을 외견상 가시화되도록 기술하는 방법이다. 사용사례로 얻은 요구사항 문서는 사용자 훈련 매뉴얼, 수락시험 기준 또는 기능점수 계산에 사용될 수 있다.

2.4 기능 기반 소프트웨어 규모 측정 기법들의 장·단점

FPA의 방법은 그 목적에 따라 3가지 그룹으로 대별 할 수 있는데, 첫째 그룹은 계측 규칙을 엄밀화 하여 계측 오차의 극소화를 목표로 하며, 대표적인 기법이 IFPUG 법이다. IFPUG 법의 개발 목적은 누가 소프트웨어 규모 계측을 하여도 동일한 값이 산출 되도록 계측 방법을 치밀하게 정의하려고 하는 것이며, 이를 위해 측정 수행 지침(Counting Practices Manual 2,3,4)이 제공되고 있다. 하지만, 현재 제공되는 영문 4.1판에도 정의나 규칙 누락 또는 모호성이 지적되고 있다.

둘째 그룹은 계측 방법을 가능한 한 간소화하여 개발 초기 단계에서의 적용을 목표로 하는 것이다. 이 그룹의 기법은 사양의 상세가 정해지지 않아도 간단하게 측정을 할 수 있기 때문에 측정치의 정밀성 보다 신속성과 간편함이 요구되는 초기 공정에서 적용하는 경우가 많다. 예를 들어 SPR 개산법[36]에서는 일부만을 계측한 결과를 가지고 전체의 기능량을 추정한다.

셋째 그룹은 엄밀성은 위 2번째 그룹의 중간이지만, 개발 현장에의 도입 용이성을 중시하는 것이다. 이는 국제 통일 규칙에 기초한 글로벌 척도는 아니지만 자기조직에 맞는 로컬한 척도로 좋기 때문에 도입이 용이하고 계측하기 쉬운 기법을 지향한다.

[표 9]는 위 3가지 그룹 중에서 첫 번째 그룹인 소프트웨어 규모 측정의 규칙을 엄밀히 하여 측정값의 오차 최소화를 목표로 하는 그룹의 기능점수법의 종류와 각각의 장단점을 나타낸다[37].

[표 9] 기능점수법의 파생 기법의 장단점

기능점수 기법	적용대상분야		적용가능시기		계측 노력	계측정밀도		도입용이성		
	사무 처리	장비내재/ 제어	기본설계 착수전후	기본설계 종료 후		초심자가 계측할때	숙련자가 계측할때	도입시 교육의 용이성	계측자의 습득의 용이성	타조직 에의 이식성
IBM법	◎	×	△	◎	약간多	○	○	○	약간難	高
IFPUG법	◎	×	△	◎	매우多	△	◎	×	매우難	高
전중연법(제2판)	◎	×	◎	◎	약간多	○	◎	○	약간難	高
FPA Mark-Ⅱ)	◎	×	△	◎	약간多	○	○	△	약간難	低
3DFPs	◎	○	△	◎	매우多	△	◎	×	매우難	약간低
Boeing 3D Ext.	◎	○	△	◎	매우多	△	◎	×	매우難	低

◎ : 매우 적합, ○ : 적합, △ : 보통, × : 부적합

[표 9]에서는 각 기법에 대하여, 적용 대상 분야, 적용 가능 시기, 계측 노력, 계측 정밀도, 도입 용이성의 5개 분야에서 평가 하였다. 각 기법은 각각의 장단점이나 특정 목적에 따라 다르므로 어떤 기법이 좋고 어떤 기법이 나쁘다고는 단정 할 수 없다. 분야별 평가 내용은 다음과 같다[37].

첫째, 적용 대상 분야의 비교는 FPA가 사무처리 분야 중심으로 발전해 온 것이기 때문에 모든 기법이 사무처리 분야의 적용이 가능하다. 장비내재/제어 분야에서는, 적용 대상 확대의 시도를 하고 있는 것은 3개 기법 뿐이며, 이 모두가 아직 계측자의 노하우에 의존한 부분이 많아, 범용성이라

고 하는 의미에서는 아직 실용단계에 이르지 못하고 있다.

둘째, 적용 가능 시기와 계측 정밀도의 비교는 FPA에 의한 기능량 계측치의 정밀도는 계측자의 주관적 판단 차이에 의한 값의 차이 발생이나, 계측시의 누락이나 오판정에 의한 값의 차이 발생 또는 하나 하나 세는 방법의 차이에 의한 값의 차이 발생과 같이 3가지 종류의 요인에 의해 결정된다. 이 3개의 관점에서 기법을 평가하면 적용 가능 시기와 계측 정밀도는 Trade Off 관계에 있다. 다시 말해, 공수견적을 공정의 초기단계에서 적용하고자 한다면, 자연히 결정되지 않은 사항들이 많아 계측치의 정밀도는 낮아지게 된다.

셋째, 계측 노력과 계측 정밀도의 비교는 Trade Off 관계에 있다. 계측치의 오차를 최소화하기 위해서는 개인의 주관적 판단여지를 매우 적게 할 필요가 있지만 객관성을 높이기 위해서는 체크리스트를 이용 하든지 요소수를 세는 수순이 필요하게 되며, 이에 따라 계측 노력도 증가하게 된다. IBM 법이나 IFPUG 법의 경우 항목의 수를 세는 작업을 의무화 하고 있어, 이 작업에 소요되는 시간이 전체 계측 시간의 50%이상을 소비하는 경우가 많다.

넷째, 도입 용이성의 비교는 도입 용이성 측면에서 정밀 계측 기법은 습득에 많은 시간이 소요된다. 특정한 개발 기법이나 개발 수순에 의존한 계측 방법을 취하고 있는 것은 타 조직에 이식이 곤란하기 때문에, 경우에 따라서는 그대로 도입한 후 커스터마이즈를 하는 편이 전체적인 도입 비용을 절감 할 수 있다[38].

3. 기능점수 및 COSMIC-FFP 측정

본장에서는 실무사례연구로써 통합행정업무시스템을 Counting Practices Manual 4.1과 COSMIC-FFP V.3.0 매뉴얼에 따라 기능점수와 완전기능점수를 직접 산정해 본다.

3.1 실무 사례 적용환경

이 절에서는 실무 사례 연구로 수행된 개발 프로젝트의 구성 및 각 서브시스템에 대해 소개하고, 시스템 개발을 위한 환경 및 투입인원, 개발기간에 대해 설명한다. 통합행정업무 시스템의 개발 기간은 총 4명의 인원이 요구사항 분석에서 구현까지 5개월의 기간 동안 투입 되어 수행되었으며, 개발공정 중 시험은 이 기간에서 제외되었다. 프로젝트의 기능점수는 각각의 서브시스템별로 계산되었으며, 실제 프로젝트 투입시간 또한 서브시스템별/공정별로 측정되었다. 프로젝트에 참여한 모든 구성원은 한국소프트웨어산업협회에서 제시하는 기술자 등급 및 자격기준에 따라 기사 10년 이상의 특급기술자로 구성되었으며, 개발 플랫폼은 Windows의 웹기반으로 개발언어는 ASP(Active Server Page), DBMS는 MS-SQL 2000, 웹서버는 IIS(Internet Information Server), 서버 운영체제는 Windows 2000 서버로 구성된다. 하드웨어는 Dell사의 Poweredge-SC1 420 Server로 메모리 1G, SCSI Hard 146G Dual, Xeon(TM) Processor 2.8GHz 사양으로 구성된다. 모든 프로그램 사용자는 웹을 통해서 접근하며, 각 서브시스템의 기능별로 사용자 권한을 부여 하여 개인별 접근 내용이 차별화 된다. 통합행정업무 시스템은 총 8개 서브시스템으로 구성되며, 새로운 시스템 개발을 목적으로 하며, 각 서브시스템별 개요는 다음과 같다.

첫째, 예산관리 시스템은 예산 항목별로 예산 편성, 배정 및 자금 배정과 반납의 기능을 가지며, 연도별 예산에 대한 집행현황 및 예산서를 관리한다. 둘째, 수입관리 시스템은 수입과 관련하여 발생되는 모든 결의서를 생성하며, 승인된 결의서에 대한 통계자료를 관리한다. 셋째, 계약관리 시스템은 지출과 관련하여 발생되는 모든 결의서를 생성하며, 법인카드 및 수탁관련 자료를 관리하며, 승인된 결의서에 대한 통계자료를 생성한다. 넷째, 회계관리 시스템은 수입과 계약관리 시스템으로부터 요청된 결의서에 대해 승인 또는 반려를 하며, 예산대비 지출 내역에 대한 통계자료를 생성한다. 다섯째, 인사관리 시스템은 인사정보, 인사고과, 인사발령을 관리하며, 각종 증명서 출력 및 구성원에 대한 출신교별, 직급별 등 통계자료를 관리한다. 여섯째, 급여관리 시스템은 급여 및 수당에 대한 기본정보를 관리하고, 월별 급여 및 수당 계산과, 급여내역서 조회 및 연말정산, 퇴직금, 연가보상비를 계산한다. 일곱째, 총무관리 시스템은 출장관리, 복무관리, 차량관리, 상조회관리, 동호회관리, 우편물관리의 기능을 가지며, 각각은 독립적으로 수행된다. 여덟째, 연구실적관리 시스템은 연구과제관리, 일정관리, 자료관리, 출판물관리, 위원회관리로 구성되며, 연구과제 DB 구축을 통해 연구실적에 대한 통계자료를 생성한다.

3.2. 실무 사례 적용을 통한 기능점수 측정

본 논문의 통합행정업무 시스템은 총 8개의 서브시스템으로 구성 되며, 개발 범위는 서브시스템 모두를 대상으로 한다. 기능점수 산출과정은 Function Point Users Group Practices Manual Release 4.1에 따른 것이다. 예산관리, 계약관리, 회계관리, 인사관리, 급여관리, 총무관리, 연구실적

관리 총 8개의 서브시스템별로 기능점수를 계산하였다. 본 절에서는 예산 관리 시스템을 대상으로 기능점수 산정 과정을 기술하고, 최종적으로 각 서브 시스템별 보정 후 기능점수를 도출한다.

가. 보정전 기능점수 측정

이 절에서는 [표 10], [표 11], [표 12], [표 13]과 같이 사용자 상세 요구사항 명세를 기반으로 하여 통합행정업무 시스템의 서브시스템별 개발예상 기능 목록을 작성한다. 이 목록을 통하여 ILF, EIF, EI, EO, EQ의 개수를 파악한다. 이는 보정전 기능점수 계산의 기초 자료로 사용된다.

[표 10] 예산관리 시스템 개발예상 기능 목록표

연번	대분류	중분류	처리기능	외부입력(EI)			EO	EQ	ILF	EIF
				입력	수정	삭제	출력	조회	내부	외부
1	예산	예산 정보	예산정보관리	1	1	1			예산, 예산 집행, 예산 편성, 예산 배정, 자금 배정, 예산 반납	지출
2			예산정보조회					1		
3		예산 편성	예산편성관리	1		1				
4			예산편성열람					1		
5		예산 배정	예산배정관리	1	1	1				
6			예산배정열람				1			
7		자금 배정	자금배정관리	1	1	1				
8			자금배정열람				1			
9		예산 반납	예산반납관리	1	1	1				
10			예산반납열람				1			
11		예산집 행현황	예산집행현황 조회				1			
소계				5	4	5	4	2	6	1

[표 11] 수입관리, 계약관리, 회계관리 시스템 개발예상 기능 목록표

연번	대분류	중분류	처리기능	외부입력(EI)			EO	EQ	ILF	EIF
				입력	수정	삭제	출력	조회	내부	외부
12	수입	수입결의정보	수입결의 정보관리	1	1	1			수입결의, 납부자, 기금, 승인요청	승인
13			수입결의 승인요청	1						
14			수입결의정보 조회					1		
15			각종 보고서 조회					1		
소 계				2	1	1	0	2	4	1
16	계약	용역관리	용역기초 정보관리	1	1	1			지출결의, 용역, 법인카드, 출장, 승인요청	예산, 급여, 회계
17			용역기초정보조회					1		
18		지출경의관리	지출결의 정보관리	1	1	1				
19			지출결의 정보조회					1		
20			승인요청	1						
21		법인카드관리	법인카드사 용내역관리	1	1	1				
22			법인카드사 용내역조회					1		
23			법인카드사 용내역지출				1			
24		각종보고서관리	각종보고서 조회				1			
소 계				4	3	3	2	3	5	3
25	회계	결의서 관리	결의서 승인	1					회계	예산, 지출결의, 수입결의
26			각종보고서 조회 및 출력				1			
소 계				1	0	0	1	0	1	3

[표 12] 인사관리, 급여관리 시스템 개발예상기능 목록표

연번	대분류	중분류	처리기능	외부입력(EI)			EO	EQ	ILF	EIF
				입력	수정	삭제	출력	조회	내부	외부
27	인사	인사정보관리	인사기초정보관리	1	1	1			인사정보, 인사고과, 인사발령	급여, 연구실적
28			인사정보조회					1		
29		인사발령관리	인사발령사항관리	1	1	1				
30			인사발령사항조회					1		
31		인사고과관리	근무평정관리	1	1	1				
32			근무평정조회					1		
33		증명서관리	증명서조회				1			
34			증명서발급대장조회					1		
35			각종보고서조회				1			
소 계				3	3	3	2	4	3	2
36	급여	급여기본정보	급여기초정보관리	1	1	1			급여정보, 개인공제, 급여, 연말정산, 퇴직금, 연가보상, 수당, 수당기본정보	인사정보, 지출, 초과근무
37			급여기초정보조회					1		
38		수당정보관리	수당기본정보관리	1	1	1				
39			수당계산	1						
40			수당내역조회				1			
41			수당기본정보조회					1		
42		급여산정관리	급여기본정보관리관리	1	1	1				
43			급여계산	1						
44			급여조정	1						
45			급여소급	1						
46			급여지급내역조회				1			
47			은행별입금내역조회				1			
48		급여통계관리	연말정산조회					1		
49			퇴직금조회					1		
50			연가보상비조회					1		
소 계				7	3	3	3	5	8	3

[표 13] 총무관리, 연구실적관리 시스템 개발예상 기능 목록표

연번	대분류	중분류	처리기능	외부입력(EI)			EO	EQ	ILF	EIF
				입력	수정	삭제	출력	조회	내부	외부
51	총무 관리	출장 관리	출장정보관리	1	1	1			출장자, 출장 내역, 초과 근무, 차량, 차량 운행 정보, 차량 수리 정보, 상조회, 동호회	수당, 근무 상황
52			출장사항조회					1		
53		근무 관리	초과근무관리	1	1	1				
54			초과근무조회					1		
55			근무상황별 건수조회				1			
56										
57		차량 관리	차량정보관리	1	1	1				
58			차량정보조회					1		
59		상조회 관리	상조회기본정보관리	1	1	1				
60			상조회열람				1			
		동호회 관리	동호회기본정보관리	1	1	1				
61			동호회열람				1			
소 계				5	5	5	3	3	8	2
62	연구실 적관리	과제 관리	과제정보관리	2	1	1			연구 실적, 일정, 자료함, 출판물	인사, 정보 지출
63			과제정보 조회					1		
64			과제예산 조회				1			
65			과제실적통계 조회				1			
66		자료함 관리	자료함정보관리	1	1	1				
67		일정 관리	일정정보관리	1	1	1				
68			일정 조회					1		
69		출판물 관리	출판물정보관리	1	1	1				
70			배부처관리	1	1	1				
71			출판등록내용 조회					1		
72			배부처 조회					1		
소 계				6	5	5	2	4	4	2
합 계				82			17	23	39	17

[표 14]와 같이 보정전 기능점수는 개발예상 기능 목록을 통해 산출되어진 ILF, EIF, EI, EO, EQ의 개수에 각각의 기능 복잡도와 가중치를 부여함으로써 산출 할 수 있으며, [표 14]는 예산관리 시스템의 보정전 기능점수를 나타낸 것이다.

[표 14] 예산관리 시스템의 기능점수

기능유형	기능적 복잡도		가중치	복잡도별 합	기능유형별 합
	개수	복잡도			
ILF	6	낮음	*7 =	42	
	0	보통	*10 =	0	
	0	높음	*15 =	0	
					42
EIF	1	낮음	*5 =	5	
	0	보통	*7 =	0	
	0	높음	*10 =	0	
					5
EI	12	낮음	*3 =	36	
	0	보통	*4 =	0	
	2	높음	*6 =	12	
					48
EQ	2	낮음	*3 =	6	
	0	보통	*4 =	0	
	0	높음	*6 =	0	
					6
EO	3	낮음	*4 =	12	
	1	보통	*5 =	5	
	0	높음	*7 =	0	
					17
기능점수					118

[표 15]는 앞에서 설명한 기능점수에 기반한 소프트웨어 규모 측정 방법에 따라 서브시스템별로 기능성과 복잡도에 기초하여 계산한 보정전 기능점수를 보여준다.

[표 15] 보정전 서브시스템별 기능점수

시스템명	예산 관리	수입 관리	계약 관리	회계 관리	인사 관리	급여 관리	총무 관리	연구실적관리	총합계
기능 점수	118	63	118	31	107	158	172	116	883

나. 보정계수 측정

이 절에서는 한국소프트웨어산업협회에서 제공되는 소프트웨어사업대가의 기준에 제시된 4가지 보정계수 산출 방법[39]에 따라 보정계수를 결정하였다. 첫째, 규모 보정계수는 개발규모가 증가하면 투입인원이 증가되고 프로젝트 관리 및 투입 인력간 의사소통의 증가로 생산성이 감소되는 것을 반영하는 보정계수이다. [표 16]과 같은 방법에 따라 보정계수를 계산한다.

[표 16] 규모 보정계수 산정 방법

기능점수	규모 보정계수
300미만	0.65
300이상	0.108 * $\log_e$(기능점수) + 0.2229

따라서 본 사례연구에서는 산출된 기능점수 값이 300이상 이므로, 식(12)와 같이 기능점수 883을 대입하여 0.9555 값을 구하였다.

$$\text{규모보정계수} = 0.108 \times \log_e(883) + 0.2229 \qquad (12)$$

둘째, 어플리케이션유형 보정계수는 로직의 복잡성과 시스템의 영향 정도에 따라 생산성이 증감되는 것을 반영하는 보정계수이다. [표 17]과 같이 어플리케이션 유형에 따라 보정 계수를 달리 적용한다. 만약, 어플리케이션 유형 보정계수는 동일 소프트웨어 사업에 어플리케이션 유형이 2개 이상일 경우 구분하여 적용한다. 그리고, [표 18]은 어플리케이션 유형에 대한 분류 기준을 예를 들어 표시하였다.

[표 17] 어플리케이션유형 보정계수

어플리케이션유형	보정 계수
업무처리용	1.0
과학기술용	1.2
멀티미디어용	1.3
지능정보용	1.7
시스템용	1.7
통신제어용	1.9
공정제어용	2.0
지휘통제용	2.2

[표 18] 어플리케이션유형 분류 기준예시

어플리케이션 유형	범 위
업무처리용	인사, 회계, 급여, 영업 등 경영관리 및 업무처리용 소프트웨어 등
과학기술용	과학계산, 시뮬레이션, 스프레드시트, 통계, OR, CAE 등
멀티미디어용	그래픽, 영상, 음성, 등 멀티미디어 응용분야, 지리정보시스템, 교육· 오락용
지능정보용	자연어처리, 인공지능, 전문가시스템
시스템용	운영체제, 언어처리 프로그램, DBMS, 인간·기계 인터페이스, 윈도시스템, CASE, 유틸리티 등
통신제어용	통신프로토콜, 에뮬레이션, 교환기소프트웨어, GPS 등
공정제어용	생산관리, CAM, CIM, 기기제어, 로봇제어, 실시간, 내장형 소프트웨어 등
지휘통제용	군, 경찰 등 군장비·인력의 지휘통제를 요하는 소프트웨어

어플리케이션유형 보정계수 또한 어플리케이션유형 분류기준에 따라 통합행정업무시스템은 업무처리용에 해당된다. 따라서, 어플리케이션유형 보정계수는 0.1의 값을 가진다. 셋째, 언어 보정계수는 프로젝트 개발언어별 적용비율을 도출하여, 각 개발언어별 보정계수를 적용비율로 곱하여 언어별 보정치를 구한다. 예를 들어, 프로젝트 개발 시 개발언어를 Java, C, HTML을 각각 60%, 20%, 20% 사용한다고 가정하면, 식(13)과 같이 계산된다.

$$\text{언어 보정계수}(1.12) = 0.6 \times 1.2 + 0.2 \times 1.2 + 0.2 \times 0.8 \quad (13)$$

그리고, 개발언어 보정은 구현 및 시험 단계에서만 적용한다. 만약 개발 프로젝트 전체공정 중에서 구현 및 시험 단계가 차지하는 비중이 57%이고, 분석 및 설계가 43%이면 식(14)와 같이 계산된다.

$$\text{적용보정계수} = 0.43 + 0.57 \times 1.2(\text{언어 보정계수}) \qquad (14)$$

언어 보정계수는 개발언어에 따라 생산성이 영향 받는 점을 반영하는 보정계수이다. [표 19]는 개발언어별로 적용되는 언어 보정계수를 표시하고 있다.

[표 19] 언어 보정계수

언어구분	보정계수
Assembly, 기계어, 자연어	1.9
C, CHILL, C++, Java, C#, PROLOG, UNIX Shell Scripts	1.2
Cobol, FORTRAN, PL/1, Pascal, Ada	1.0
ABAP4, Delphi, HTML, PB, VB, Program Generator, Query default, Small Talk, SQL, Statistical default, XML default, Script default(JPS, ASP, PHP 등)	0.8
EXCEL, Spreadsheet default, Screen painter default	0.6

통합행정업무시스템은 ASP를 이용해 시스템 전체가 개발 되었으므로, 언어 보정계수는 0.8의 값을 가지게 된다. 그리고, 구현 및 시험 단계가 차지하는 비중을 계산하면 14%이고, 분석 및 설계가 86%이다. 이는 언어보정

계수가 적용되는 구현 및 시험 단계 중에서 시험 단계가 제외되어 진행되었기 때문에 이와 같은 결과를 보인다. 하지만 이 비율을 본 프로젝트에 그대로 적용하면 개발언어보정에 대한 오차가 너무 커짐으로 일반적으로 사용되는 구현 및 시험 단계의 비중을 57%로 적용 하였다. 따라서 식(15)와 같이 계산하여 0.886의 언어 보정계수를 구하였다.

적용보정계수(0.8)= 0.43 + 0.57 × 0.8(언어 보정계수) (15)

넷째, 품질 및 특성 보정계수는 일반 시스템 특성(GSC) 14개 중에서 분산 데이터 처리, 시스템 성능, 운영 용이성, 다중 설치성을 포함하고 있다. 만약, 개발 프로젝트에서 서브시스템별로 품질 및 특성 보정요소의 특성이 다르면 서브시스템별로 다른 보정치를 적용 하여야 한다.
[표 20]과 같이 품질 및 특성 보정계수는 분산처리, 성능, 신뢰도, 다중 사이트와 같이 4가지 종류의 보정요소로 구분하며, 각각은 0에서 2까지 3등급의 영향도로 가진다. 식(16)과 같은 방법에 따라 총영향도를 계산하며, 산출된 총영향도를 대입하여 식(17)과 같이 품질 및 특성 보정계수를 계산한다.

총 영향도 = 분산처리 영향도 + 성능 영향도 + 신뢰도 영향도 + 다중사이트 영향도 (16)

품질 및 특성 보정계수 = 0.025 * 총영향도 + 1 (17)

[표 20] 품질 및 특성 보정계수

보정요소		판단기준	영향도
분산처리	어플리케이션이 구성요소간에 데이터를 전송하는 정도	분산처리에 대한 요구사항이 명시되지 않음	0
		클라이언트/서버 및 웹 기반 어플리케이션과 같이 분산 처리와 자료 전송이 온라인으로 수행됨	1
		어플리케이션상의 처리기능이 복수개의 서버 또는 프로세서상에서 동적으로 상호 수행됨	2
성능	응답시간 또는 처리율에 대한 사용자요구수준	성능에 대한 특별한 요구사항이나 활동이 명시되지 않으며, 기본적인 성능이 제공됨	0
		응답시간 또는 처리율이 피크타임 또는 모든 업무시간에 중요함. 연동 시스템의 처리 마감시간에 대한 제한이 있음.	1
		성능 요구사항을 만족하기 위해 설계 단계에서부터 성능 분석이 요구되거나, 설계·개발·구현 단계에서 성능 분석 도구가 사용됨	2

[표 20] 계속

보정요소		판단기준	영향도
신뢰성	장애시 미치는 영향의 정도	신뢰성에 대한 요구사항이 명시되지 않으며, 기본적인 신뢰성이 제공됨	0
		고장시 쉽게 복구가능한 수준의 약간 불편한 손실이 발생함.	1
		고장시 복구가 어려우며, 재정적 손실이 많이 발생하거나, 인명피해 위험이 있음	2
다중 사이트	상이한 하드웨어와 소프트웨어 환경을 지원하도록 개발되는 정도	설계 단계에서 하나의 설치 사이트에 대한 요구사항만 고려됨. 어플리케이션이 동일한 하드웨어 또는 소프트웨어 환경하에서만 운영되도록 설계됨	0
		설계 단계에서 하나 이상의 설치 사이트에 대한 요구사항이 고려됨. 어플리케이션이 유사한 하드웨어 또는 소프트웨어 환경하에서만 운영되도록 설계됨	1
		설계 단계에서 하나 이상의 설치 사이트에 대한 요구사항이 고려됨. 어플리케이션이 상이한 하드웨어 및 소프트웨어 환경하에서 동작하도록 설계됨	2

통합행정업무시스템에서는 [표 21]과 같이 4가지 보정요소의 영향도를 합하여 총영향도 값 1을 구하였다.

[표 21] 통합행정업무시스템의 총영향도

보정 요소	판단기준	영향도
분산처리	클라이언트/서버 및 웹 기반 어플리케이션과 같이 분산처리와 자료전송이 온라인으로 수행됨	1
성능	성능에 대한 특별한 요구사항이나 활동이 명시되지 않으며, 기본적인 성능이 제공됨	0
신뢰성	신뢰에 대한 요구사항이 명시되지 않으며, 기본적인 신뢰성이 제공됨	0
다중사이트	설계단계에서 하나의 설치 사이트에 대한 요구사항만 고려됨. 어플리케이션이 동일한 하드웨어 또는 소프트웨어 환경하에서만 운영이 되도록 설계됨	0
총영향도	분산처리 + 성능 + 신뢰도 + 다중사이트	1

따라서, 품질 및 특성 보정계수는 식(18)과 같이 총영향도 1을 식에 대입함으로써 1.025의 값을 산출하였다.

$$\text{품질 및 특성 보정계수}(1.025) = 0.025 \times (1) + 1 \qquad (18)$$

마지막으로, 총 보정계수는 식(19)와 같이 규모 보정계수, 어플리케이션 유형 보정계수, 언어 보정계수, 품질 및 특성 보정계수 곱하여 산출한다.

총 보정계수 = 규모보정계수 × 어플리케이션유형보정계수 × 언어보정계수 × 품질및특성보정계수 (19)

총 보정계수는 식(20)과 같이 규모보정계수, 어플리케이션 유형보정계수, 개발언어보정계수, 품질 및 특성 보정계수 순서로 곱하여 산출하였다.

총 보정계수(0.7835) = 0.9555 × 1 × 0.8 × 1.025 (20)

[표 22]는 통합행정업무 시스템의 총 보정계수, 규모 보정계수, 어플리케이션 유형 보정계수, 개발언어 보정계수, 품질 및 특성 보정계수를 표시하였다.

[표 22] 보정 계수

보정 항목	보정 계수
규모	0.9555
어플리케이션 유형	1
개발언어	0.8
품질 및 특성	1.025
총 보정계수	0.7835

다. 보정후 기능점수 측정

보정전 기능점수에 총 보정계수를 곱함으로써 [표 23]과 같은 보정 후 기능점수를 산출 하였다.

[표 23] 보정후 서브시스템별 기능점수

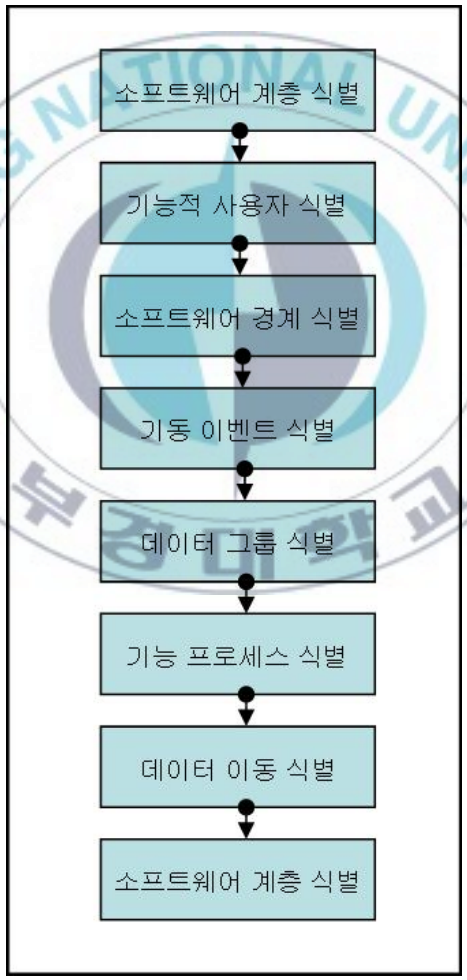
시스템명	예산관리	수입관리	계약관리	회계관리	인사관리	급여관리	총무관리	연구실적관리	총합계
기능점수	92	49	92	24	84	124	135	91	691

3.3. 실무 사례 적용을 통한 COSMIC-FFP 측정

본 절에서는 COSMIC-FFP V.3.0 매뉴얼에 따라 통합행정업무 시스템의 완전기능점수를 직접 산정해 본다. 통합행정업무 시스템은 총 8개의 서브시스템으로 구성 되며, 개발 범위는 서브시스템 모두를 대상으로 한다. 예산관리, 계약관리, 회계관리, 인사관리, 급여관리, 총무관리, 연구실적관리 총 8개의 서브시스템별로 완전기능점수를 계산한다. 본 절에서는 예산관리 시스템을 대상으로 완전기능점수 산정 과정을 기술하고, 최종적으로 각 서브 시스템별 완전기능점수를 도출한다.

가. COSMIC-FFP 측정

본 절에서는 [그림 7]과 같이 COSMIC-FFP 측정 순서에 따라 통합행정업무 시스템의 완전기능점수를 산정한다.



[그림 7] COSMIC-FFP 측정 순서

첫째, 기능 규모 측정(FSM, Functional Size Measurement) 목적과 범위를 결정한다. 측정 목적은 개발자가 만들어야 될 통합행정업무 시스템의 COSMIC-FFP 규모를 결정하는 것이며, 또한 사용자 요구사항 명세서에 따라 기능적 사용자 요구의 크기를 측정하는 것이다. 그리고, 사례연구를 통해 COSMIC-FFP 측정 방법을 업무처리 지원 시스템에 직접 적용해 소프트웨어 규모를 측정해 보는 것에 목적이 있다. 그리고, 측정 범위는 통합행정업무 시스템을 구성하고 있는 예산관리, 수입관리, 계약관리, 회계관리, 인사관리, 급여관리, 총무관리, 연구실적관리 총 8개의 서브시스템에 대한 사용자 요구사항에 할당된 모든 기능을 대상으로 한다. 단, 이후 기술되는 모든 내용은 8개의 서브시스템 중 예산관리 시스템을 대상으로 예를 들어 설명한다.

둘째, 기능적 사용자 요구(FUR, Functional User Requirements) 식별은 통합행정업무 시스템의 기능적 사용자 요구를 다음과 같이 식별 하였다. 본 논문의 통합행정업무 시스템은 총 8개의 서브시스템으로 구성 되며, 개발 범위는 서브시스템 모두를 대상으로 한다. 아래는 각 서브시스템별로 구체적인 요구사항을 기술하여, COSMIC-FFP 측정의 기초 자료로 활용한다. 예산관리 시스템의 기능적 사용자 요구(FUR)은 [표 24]와 같다

[표 24] 예산관리 시스템 기능적 사용자 요구(FUR)

단위 시스템명	기능적 사용자 요구	내용
예산정보 관리	FUR1 예산에 대한 기초 정보	주) 한 개의 예산항목은 세입/세출, 회계구분, 장, 관, 항, 목, 세목의 기초자료를 입력 받는다.
	FUR2 예산정보 수정	예산정보 수정은 세입/세출을 제외하고 나머지 항목에 대해 수정이 가능하다.
	FUR3 예산정보 삭제	예산정보 삭제는 세입/세출을 제외하고 나머지 항목에 대해 삭제가 가능하다.
	FUR4 예산정보 조회	장, 관, 항, 목, 세목으로 구성된 예산코드 및 예산명으로 검색하여 예산목록을 모두 보여 준다.
예산편성 관리	FUR1 예산편성에 관한 기본 정보	주)예산편성은 매년초에 편성을 하며, 추경 발생 시 추경에 대한 예산편성이 다시 이루어진다 . 편성 일자 별로 입력 하며 일자는 한번 선택하면 변경하기 전까지 그 일자를 유지한다.
	FUR2 예산편성 삭제	세목을 기준으로 입력된 예산편성 금액이 잘못 입력되었을 경우 편성금액을 삭제 한다. 별도의 수정은 가능은 가지지 않으면 삭제 후 다시 예산 편성을 한다.
	FUR3 예산편성 열람	주) 예산편성 열람은 편성일자를 기준으로 조회되며, 세목을 검색하여 예산 편성 내용을 조회하고 출력 가능하게 한다.

[표 24] 계속

단위 시스템명	기능적 사용자 요구	내용
예산배정 관리	FUR1 예산배정에 관한 기본 정보	주)예산배정은 예산이 편성된 예산과목과 금액 범위 내에서 배정이 이루어지며, 분기단위로 예산 배정은 이루어진다. 예산배정액은 금액 입력이나 예산편성액에 대한 % 입력 2가지 방법으로 처리 할 수 있어야 한다. 일괄처리를 가능을 두어 일괄처리 할 경우 예산편성액의 전부가 예산배정액으로 처리 되어 분기별 배정이 필요 없게 된다. 회계시스템으로부터 실제 지출액을 예산과목별로 가져와 현재까지 지출금액을 표시 해준다.
	FUR2 예산배정 수정	배정일자를 기준으로 입력된 예산배정 금액이 잘못 입력되었을 경우 배정금액을 수정 한다. 배정일자를 기준으로 분기별 배정금액을 분류한다.
	FUR3 예산배정 삭제	배정일자를 기준으로 입력된 예산배정 금액이 잘못 입력되었을 경우 예산배정에 관련된 모든 정보를 삭제 한다.
	FUR4 예산배정 열람	주) 검색결과에서 예산편성금액 - 기 배정금액 = 배정잔액, 실 지출액 = '회계시스템'에서 '지출결의서'로 지출이 확정된 세목의 항목들의 합 즉, 배정 받은 예산을 실제로 얼마만큼 사용했는가를 보여주는 항목 예산배정 열람은 배정일자를 기준으로 조회되며, 세목을 검색하여 예산배정 내용을 조회하고 출력 가능하게 한다.
자금배정 관리	FUR1 자금배정에 관한 기본 정보	주)자금배정은 예산이 배정된 금액 범위 내에서 자금배정이 이루어지며, 분기단위로 자금 배정은 이루어진다. 일괄처리를 가능을 두어 일괄처리 할 경우 예산배정액의 전부가 자금배정액으로 처리 되어 분기별 자금배정이 필요 없게 된다. 회계시스템으로부터 실제 지출액을 예산과목별로 가져와 현재까지 지출금액을 표시 해준다.
	FUR2 자금배정 수정	배정일자를 기준으로 입력된 예산배정 금액이 잘못 입력되었을 경우 배정금액을 수정 한다. 배정일자를 기준으로 분기별 배정금액을 분류한다.
	FUR3 자금배정 삭제	배정일자를 기준으로 입력된 자금배정 금액이 잘못 입력되었을 경우 자금배정에 관련된 모든 정보를 삭제 한다.
	FUR4 자금배정 열람	주)자금배정 열람은 배정일자를 기준으로 조회되며, 세목을 검색하여 자금배정내용을 조회하고 출력 가능하게 한다.

[표 24] 계속

단위 시스템명	기능적 사용자 요구	내용
예산반납 관리	FUR1 예산반납에 관한 기본 정보 예산년도를 입력	주)예산반납은 자금이 배정된 금액 범위 내에서 분기별로 실제 지출이 발생된 금액을 제외하고 자금배정이 남으면 예산을 반납하게 되고 반납된 예산은 다음분기의 자금배정 금액에 합산되어진다. 일괄처리 기능을 두어 일괄처리 할 경우 자금 배정액 중에서 실제 지출 후 남은 금액에 대하여 모든 예산과목에 적용하여 일괄적으로 예산반납을 실행한다. 회계 시스템의 실제 지출액을 비교한다.
	FUR2 예산반납 수정	자금배정일자를 기준으로 입력된 예산반납 금액이 잘못 입력되었을 경우 반납금액을 수정 한다. 회계 시스템의 실 지출액을 자금배정액에서 차 함으로써 자동 계산되어진다. 자금배정일자를 기준으로 분기별 예산반납금액을 분류한다.
	FUR3 예산반납 삭제	자금배정일자를 기준으로 입력된 예산반납 금액이 잘못 입력되었을 경우 반납에 관련된 모든 정보를 삭제 한다.
	FUR4 예산반납 열람	주)예산반납에 대한 분기별 내역 및 연도별 총괄 내역을 조회한다.
예산집행현황조회	FUR1 연도별 예산집행 현황 조회	분기별 자금배정일자 별로 집행현황을 조회한다. 실제 지출금액은 회계 시스템에서 유지된다.

■ COSMIC-FFP 측정 과정

① 소프트웨어 계층 식별

소프트웨어는 기능적으로 운영되는 환경 내의 기능 추상화 수준이 다른 여러 요소를 가질 수 있는데, 이들 요소 간의 기능 교환에 기초한 추상화 수준 사이에는 계층적 관계가 있다. COSMIC-FFP 측정 매뉴얼 3.0[29]에 따라 통합행정업무 시스템의 모든 기능적 사용자 요구는 단일 소프트웨어 요소에 할당됨으로 소프트웨어 계층은 단일 계층의 구성되어 있다.

② 기능적 사용자 식별

통합행정업무 시스템과 상호작용하는 기능적 사용자가 누구인지를 식별하는 과정이다. 먼저 통합행정업무 시스템에 데이터를 전송하는 기능적인 사용자를 발견 할 수 없다. 이는 통합행정업무 시스템과 연결된 별도의 시스템이 없기 때문이다. 두 번째로 통합행정업무 시스템으로부터 데이터를 받아들이는 기능적 사용자 또한 존재하지 않음을 의미한다. 그리고, 마지막으로 시스템과 상호작용하는 인간 사용자가 있는지를 확인한다. 통합행정업무 시스템에는 예산관리, 수입관리, 계약관리 등 각각의 서브 시스템을 사용하고 있는 개별 담당자들이 시스템과 상호작용하고 있으며, 본 논문에서는 이를 대표하여 통합행정업무 시스템 사용자라고 표현 하였다.

③ 소프트웨어 경계 식별

기능 규모 측정(FSM)의 범위 안에 있는 것으로 식별된 계층 내의 개별 소프트웨어의 경계는 반드시 식별되어야 하며, 일단 경계가 식별되면, 기능 규모 측정의 범위 안에서 식별된 각 기능적 사용자 요구(FUR)는 개별 소프트웨어에 반드시 할당되어야 한다. 따라서, 기능적 사용자 요구를 기반으

로 통합행정업무 시스템의 소프트웨어 경계를 식별할 수 있다.

④ 기동 이벤트 식별

기동 이벤트는 소프트웨어 경계 밖에 있는 기능적인 사용자에 의해 발생되는 이벤트로써 이 이벤트들에 의해 하나 이상의 기능 프로세스들이 발생시킨다. 통합행정업무 시스템에서는 소프트웨어 경제 밖의 통합행정업무 시스템 사용자에 의해 다음과 같은 기동 이벤트가 발생되며, 기능적 사용자 요구(FUR)를 기반으로 식별하였다. [표 25]는 예산관리 시스템의 기동 이벤트를 나타내고 있다.

[표 25]예산관리 시스템 기동 이벤트

단위 시스템명	기동 이벤트명
예산정보관리	예산기초 정보 입력 이벤트
	예산정보 수정 이벤트
	예산정보 삭제 이벤트
	예산정보 조회 이벤트
예산편성관리	예산편성 정보 입력 이벤트
	예산편성 삭제 이벤트
	예산편성 열람 이벤트
예산배정 관리	예산배정 정보 입력 이벤트
	예산배정 수정 이벤트
	예산배정 삭제 이벤트
	예산배정 열람 이벤트
자금배정 관리	자금배정 정보 입력 이벤트
	자금배정 수정 이벤트
	자금배정 삭제 이벤트
	자금배정 열람 이벤트
예산반납 관리	예산반납 정보 입력 이벤트
	예산반납 수정 이벤트
	예산반납 삭제 이벤트
	예산반납 열람 이벤트
예산집행현황 조회	예산집행 현황 조회 이벤트

⑤ 데이터 그룹 식별

기능 규모 측정(FSM) 범위 안에서 식별된 각 데이터 그룹은 데이터 속성에 대한 고유한 수집 활동을 통해 유일해야 하고 구별이 가능해야 하며, 소프트웨어의 기능적 사용자 요구 사항 내에 기술된 관심 대상의 하나와 직접적으로 관련되어야 한다. 관심 대상은 기능적 사용자 요구 사항의 관점에서 식별되는 것으로 물리적인 것일 수도 있고, 소프트웨어로 데이터를 처리하거나 또는 처리하여 저장하도록 요구하는 사용자 세계에서의 어떤 개념적 대상 또는 그 일부일 수도 있다. 그리고, 관심 대상은 엔티티 관계 도상에서의 '엔티티 유형'과 동의어이고, 제3 정규형인 관계의 주체와 같은 의미를 가진다. 언급된 조건에 따라 통합행정업무 시스템의 데이터 그룹을 식별하면 [표 26]과 같이 나타낼 수 있다.

[표 26]예산관리 시스템 데이터 그룹

기능적 사용자	관심대상 (object of interest)	데이터 그룹	비고
- 데이터를 전송하는 기능적인 사용자			
해당사항 없음			
- 데이터를 받아들이는 기능적인 사용자			
해당사항 없음			
- 시스템과 상호작용하는 인간 사용자(기능적 프로세스 동안 지속적인 데이터 교환)			
예산관리 담당자	예산	― 예산명, 회계구분 데이터 ― 장명 데이터 ― 관명 데이터 ― 항명 데이터 ― 목명 데이터 ― 세목명 데이터 ― 예산정보 데이터	회계구분: 세입/세출 예산정보: 예산명, 장, 관, 항, 목, 세목

[표 26] 계속

가능적 사용자	관심대상 (object of interest)	데이터 그룹	비고
- 시스템과 상호작용하는 인간 사용자(가능적 프로세스 동안 지속적인 데이터 교환)			
예산관리 담당자	예산편성	— 예산편성 데이터 — 예산편성삭제 데이터 — 예산편성반납 데이터 — 예산편성조회 데이터	예산편성: 예산명, 편성일자, 예산편성금액
	예산배정	— 예산배정 데이터 — 예산편성 데이터 — 예산배정수정 데이터 — 예산배정삭제 데이터 — 예산배정조회 데이터 — 지출 데이터	예산배정: 배정일자, 편성일자, 예산명, 배정금액 지출: 예산명,지출일자, 지출금액
	자금배정	— 자금배정 데이터 — 자금배정수정 데이터 — 자금배정삭제 데이터 — 자금배정조회 데이터 — 지출 데이터	자금배정: 자금배정일자,예산배정일자,예산명, 자금배정금액
	예산반납	— 예산반납 데이터 — 자금배정 데이터 — 예산반납수정 데이터 — 예산반납삭제 데이터 — 예산반납조회 데이터 — 지출 데이터	예산반납:예산반납일자, 예산명, 예산반납금액,자금배정일자
	장	— 장명 데이터	장코드, 장명
	관	— 관명 데이터	관코드, 관명
	항	— 항명 데이터	항코드, 항명
	목	— 목명 데이터	목코드, 목명
	세목	— 세목명 데이터	세목코드, 세목명
	지출	— 지출 데이터	예산명,지출일자,지출금액

⑥ 기능 프로세스 식별

기능적 사용자 요구 사항 집합의 단위 요소로서 유일하고 응집력이 있으며 독립적으로 실행이 가능한 데이터 이동의 집합을 포함한다. 또한, 하나 이상의 기동 이벤트에 의해 직접 기동되거나 액터를 통해 간접적으로 기동되며, 기능 프로세스는 기동 이벤트에 반응하여 이루어져야 할 모든 요구 사항이 실행되어야 완료된다. 식별된 각 기능 프로세스는 최소한 하나의 식별된 기능적 사용자 요구로부터 추출되어야 하며, 기동 이벤트에 의거하여 시작되어야 한다. 그리고 들어가기 하나에 나가기 또는 쓰기와 같이 최소한 두 개의 데이터 이동을 포함해야 한다. 측정 단위로 1 CFSU를 사용하며, 단위 소프트웨어에 대한 가장 작은 이론적 기능 규모는 2 CFSU가 된다. 통합행정업무 시스템의 사용자 요구사항을 기반으로 [표 27]과 같이 기능 프로세스를 식별할 수 있다.

통합행정 시스템에서 식별된 각각의 후보 기능 프로세스를 대상으로 예제1과 같이 기능 프로세스 선정의 적정성에 대해 평가하였다.

예제1) 예산정보 등록 프로세스

- 질문 : 기동 이벤트에 의해 기동 되었는가?
  답변: 예(예산정보 입력 이벤트에 의해 기동)

- 질문 : 유일하고 응집력이 있으며 독립적으로 실행이 가능한 데이터 이동의 집합을 포함하고 있는가?
  답변 : 예

- 질문 : 기동 이벤트에 반응하여 이루어져야 할 모든 요구 사항이 실행되었을 때 완료되는가?

답변 : 예

- 질문 : 기동 이벤트에 반영하여 이루어져야 할 모든 요구사항을 수행합니까?

답변 : 예

⑦ 데이터 이동 식별

식별된 각 기능 프로세스는 구성되는 요소 데이터 이동들로 분할되어야 하며, 각 데이터 이동들은 들어가기, 나가기, 읽기 또는 쓰기 중의 하나이어야 한다. 또한 식별된 기능 프로세스 내에서 단 한번만 계산되어야 하며, 다른 데이터 이동형을 포함하지 않는다. COSMIC-FFP에는 들어가기(Entry, E), 나가기(Exit, X), 읽기(Read, R), 쓰기(Write, W) 4가지 유형의 데이터 이동이 있다. 들어가기(E)는 데이터 그룹이 사용자로부터 경계를 넘어 그것을 필요로 하는 기능 프로세스로 이동하는 데이터 이동형이다. 나가기(X)는 데이터 그룹이 기능 프로세스에서 경계를 넘어 그것을 요청한 사용자에게 이동하는 데이터 이동형이다. 읽기(R)는 데이터 이동을 요구한 기능 프로세스가 미치는 영향 범위 내의 영구 저장소로부터 데이터 그룹이 이동하는 데이터 이동형이다. 쓰기(W)는 기능 프로세스에 있는 데이터 그룹을 영구 저장소로 이동하는 데이터 이동형이다. 위와 같은 식별 기준에 따라 기능 프로세스, 기능 이벤트, 데이터 이동형 [표 27]에 표시하였다.

[표 27] 예산관리 시스템 기능 프로세스, 데이터 이동 및 데이터 그룹의 세부 내용

번호	기능 프로세스 설명	기동 이벤트	데이터 이동 설명	데이터 그룹	데이터 이동형	CFSU	총 CFSU
1	예산정보 등록	예산정보 입력 이벤트	기동 이벤트에 의한 데이터 수신	예산명, 회계구분 데이터	E	1	
			장명 수신	장명 데이터	R	1	
			관명 수신	관명 데이터	R	1	
			항명 수신	항명 데이터	R	1	
			목명 수신	목명 데이터	R	1	
			세목명 수신	세목명 데이터	R	1	
			예산정보를 DB에 저장	예산정보 데이터	W	1	
							7
2	예산정보 수정	예산정보 수정 이벤트	기동 이벤트에 의한 데이터 수신	예산명 데이터	E	1	
			장명 읽기	장명 데이터	R	1	
			관명 읽기	관명 데이터	R	1	
			항명 읽기	항명 데이터	R	1	
			목명 읽기	목명 데이터	R	1	
			세목명 읽기	세목명 데이터	R	1	
			예산정보 DB에 저장	예산정보 데이터	W	1	
							7
3	예산정보 삭제	예산정보 삭제 이벤트	기동 이벤트에 의한 데이터 수신	예산명 데이터	E	1	
			장명 읽기	장명 데이터	R	1	
			관명 읽기	관명 데이터	R	1	
			항명 읽기	항명 데이터	R	1	
			목명 읽기	목명 데이터	R	1	
			세목명 읽기	세목명 데이터	R	1	
			예산정보 DB에서 삭제	예산정보 데이터	W	1	
							7

[표 27] 계속

번호	기능 프로세스 설명	기동 이벤트	데이터 이동 설명	데이터 그룹	데이터 이동형	CFSU	총 CFSU
4	예산정보 조회	예산정보 조회 이벤트	기동 이벤트에 의한 데이터 수신	예산명 데이터	E	1	
			장명 읽기	장명 데이터	R	1	
			과명 읽기	관명 데이터	R	1	
			항명 읽기	항명 데이터	R	1	
			목명 읽기	목명 데이터	R	1	
			세목명 읽기	세목명 데이터	R	1	
			예산내역 보내기	예산정보 데이터	X	1	
							7
5	예산편성 등록	예산편성 입력 이벤트	기동 이벤트에 의한 데이터 수신	예산편성 데이터	E	1	
			예산과목 읽기	예산과목 데이터	R	1	
			예산편성내용 DB에 수정	예산편성정보 데이터	W	1	
							3
6	예산편성 삭제	예산편성 삭제 이벤트	기동 이벤트에 의한 데이터 수신	예산편성 데이터	E	1	
			예산과목 읽기	예산과목 데이터	R	1	
			예산편성내용DB에서삭제	예산편성정보 데이터	W	1	
							3
7	예산편성 열람	예산편성 열람 이벤트	기동 이벤트에 의한 데이터 수신	예산편성 데이터	E	1	
			예산과목 읽기	예산과목 데이터	R	1	
			예산반납내용 읽기	예산반납 데이터	R	1	
			예산편성정보 보내기	예산편성정보 데이터	X	1	
							4

[표 27] 계속

번호	기능 프로세스 설명	기동 이벤트	데이터 이동 설명	데이터 그룹	데이터 이동형	CFSU	총 CFSU
8	예산배정 등록	예산배정 입력 이벤트	기동 이벤트에 의한 데이터 수신	예산배정 데이터	E	1	
			예산편성 내용 읽기	예산편성 데이터	R	1	
			예산과목 읽기	예산과목 데이터	R	1	
			예산배정내용 DB에 저장	예산배정정보 데이터	W	1	
							4
9	예산배정 수정	예산배정 수정 이벤트	기동 이벤트에 의한 데이터 수신	예산배정 데이터	E	1	
			예산편성내용 읽기	예산편성 데이터	R	1	
			예산과목 읽기	예산과목 데이터	R	1	
			예산배정내용 DB에 수정	예산배정정보 데이터	W	1	
							4
10	예산배정 삭제	예산배정 삭제 이벤트	기동 이벤트에 의한 데이터 수신	예산배정 데이터	E	1	
			예산편성내용 읽기	예산편성 데이터	R	1	
			예산과목 읽기	예산과목 데이터	R	1	
			예산배정내용 DB에서삭제	예산배정정보 데이터	W	1	
							4
11	예산배정 열람	예산배정 열람 이벤트	기동 이벤트에 의한 데이터 수신	예산배정 데이터	E	1	
			예산편성내용 읽기	예산편성 데이터	R	1	
			예산과목 읽기	예산과목 데이터	R	1	
			예산배정정보보내기	예산배정정보 데이터	X	1	
			지출정보 읽기	지출 데이터	R	1	
							5

[표 27] 계속

번호	기능 프로세스 설명	기동 이벤트	데이터 이동 설명	데이터 그룹	데이터 이동형	CFSU	총 CFSU
12	자금배정 등록	자금배정 입력 이벤트	기동 이벤트에 의한 데이터 수신	자금배정 데이터	E	1	
			예산배정내용 읽기	예산배정 데이터	R	1	
			예산과목 읽기	예산과목 데이터	R	1	
			자금배정내용 DB에 저장	자금배정정보 데이터	W	1	
							4
13	자금배정 수정	자금배정 수정 이벤트	기동 이벤트에 의한 데이터 수신	자금배정 데이터	E	1	
			예산배정내용 읽기	예산배정 데이터	R	1	
			예산과목 읽기	예산과목 데이터	R	1	
			자금배정내용 DB에 수정	자금배정정보 데이터	W	1	
							4
14	자금배정 삭제	자금배정 삭제 이벤트	기동 이벤트에 의한 데이터 수신	자금배정 데이터	E	1	
			예산배정내용 읽기	예산배정 데이터	R	1	
			예산과목 읽기	예산과목 데이터	R	1	
			자금배정내용 DB에서삭제	자금배정정보 데이터	W	1	
							4
15	자금배정 열람	자금배정 열람 이벤트	기동 이벤트에 의한 데이터 수신	자금배정 데이터	E	1	
			예산배정내용 읽기	예산배정 데이터	R	1	
			예산과목 읽기	예산과목 데이터	R	1	
			자금배정내용 보내기	자금배정정보 데이터	X	1	
			지출내용 읽기	지출 데이터	R	1	
							5

[표 27] 계속

번호	기능 프로세스 설명	기동 이벤트	데이터 이동 설명	데이터 그룹	데이터 이동형	CFSU	총 CFSU
16	예산반납 등록	예산반납 입력 이벤트	기동 이벤트에 의한 데이터 수신	예산반납 데이터	E	1	
			예산과목 읽기	예산과목 데이터	R	1	
			자금배정내용 읽기	자금배정 데이터	R	1	
			예산반납내용 DB에 저장	예산반납정보 데이터	W	1	
							4
17	예산반납 수정	예산반납 수정 이벤트	기동 이벤트에 의한 데이터 수신	예산반납 데이터	E	1	
			예산과목 읽기	예산과목 데이터	R	1	
			자금배정내용 읽기	자금배정 데이터	R	1	
			예산반납내용 DB에 수정	예산반납정보 데이터	W	1	
							4
18	예산반납 삭제	예산반납 삭제 이벤트	기동 이벤트에 의한 데이터 수신	예산반납 데이터	E	1	
			예산과목 읽기	예산과목 데이터	R	1	
			자금배정내용 읽기	자금배정 데이터	R	1	
			예산반납정도 DB에서삭제	예산반납정보 데이터	W	1	
							4
19	예산반납 열람	예산반납 열람 이벤트	기동 이벤트에 의한 데이터 수신	예산반납 데이터	E	1	
			예산과목 읽기	예산과목 데이터	R	1	
			자금배정내용 읽기	자금배정 데이터	R	1	
			예산반납정보보내기	예산반납정보 데이터	X	1	
			지출내용 일기	지출 데이터	R	1	
							5

[표 27] 계속

번호	기능 프로세스 설명	기동 이벤트	데이터 이동 설명	데이터 그룹	데이터 이동형	CFSU	총 CFSU
20	예산집행조회	예산집행 열람 이벤트	기동 이벤트에 의한 데이터 수신	예산정보 데이터	E	1	
			예산편성내용 읽기	예산편성 데이터	R	1	
			예산배정내용 읽기	예산배정 데이터	R	1	
			자금배정내용 읽기	자금배정 데이터	R	1	
			예산반납내용 읽기	예산반납 데이터	R	1	
			지출내용 읽기	지출 데이터	X	1	
			예산집행현황 보내기	예산집행현황 데이터	X	1	
							7

⑧ COSMIC-FFP 규모 계산

측정 단위인 1 CFSU는 식별된 각 데이터 이동 즉, 식별된 각 데이터 이동의 사례 (들어가기, 나가기, 읽기, 쓰기)마다 할당되어야 하며, 식별된 기능 프로세스 내 모든 식별된 데이터 이동에 적용된 것으로 다음 절차에 의거하여 해당 기능 프로세스에 대한 단일 기능 규모 값으로 취합되어야 한다. 모든 데이터 이동형에 대해 단위 규모는 1이다. 먼저, 각 데이터 이동형 내 데이터 이동의 수에 해당 단위 규모를 곱한 후 각 데이터 이동형의 규모를 합산하여 기능 프로세스의 규모를 산정한다.

그 결과, 해당 기능 프로세스의 기능 규모는 식(21)을 이용하여 CFSU로 계산된다.

$$FP\ Size = \sum(Ne * Eus)i + \sum(Nx * Xus)i + \sum(Nr * Rus)i + \sum(Nw * Wus)i \qquad (21)$$

[표 28]은 예산관리 시스템을 예를 들어 기능별 프로세스별로 데이터 이동형을 표시하고 그기에 따른 완전기능점수를 표시하고 있다. 또한 [표 29]는 8개의 서브시스템별로 데이터이동형의 개수와 완전기능점수를 나타내고 있

다. [표 30]은 통합행정업무 시스템 전체에 대한 데이터 이동형별로 완전기능점수와 각각의 데이터이동 유형이 차지하고 있는 비율을 표시 하였다.

[표 28] 예산관리 COSMIC-FFP 측정 결과

번호	기능 프로세스 설명	데이터 이동 수				규모 (CFSU)
		E	X	R	W	
1	예산정보 등록	1	0	5	1	7
2	예산정보 수정	1	0	5	1	7
3	예산정보 삭제	1	0	5	1	7
4	예산정보 조회	1	1	5	0	7
5	예산편성 등록	1	0	1	1	3
6	예산편성 삭제	1	0	1	1	3
7	예산편성 열람	1	1	2	0	4
8	예산배정 등록	1	0	2	1	4
9	예산배정 수정	1	0	2	1	4
10	예산배정 삭제	1	0	2	1	4
11	예산배정 열람	1	1	3	0	5
12	자금배정 등록	1	0	2	1	4
13	자금배정 수정	1	0	2	1	4
14	자금배정 삭제	1	0	2	1	4
15	자금배정 열람	1	1	3	0	5
16	예산반납 등록	1	0	2	1	4
17	예산반납 수정	1	0	2	1	4
18	예산반납 삭제	1	0	2	1	4
19	예산반납 열람	1	1	3	0	5
20	예산집행 조회	1	2	4	0	7
합 계		20	7	55	14	96

[표 29] 서브시스템별 COSMIC-FFP

서브 시스템명	데이터 이동 수				규모 (CFSU)
	E	X	R	W	
예산관리 시스템	20	7	55	14	96
수입관리 시스템	8	5	10	3	26
계약관리 시스템	22	14	10	17	63
회계관리 시스템	6	1	4	1	12
인사관리 시스템	15	9	11	52	87
급여관리 시스템	28	9	22	27	86
총무관리 시스템	22	5	23	36	88
연구실적관리 시스템	23	6	54	16	96
합계	144	56	189	166	554

[표 30] 데이터 이동형(Type) 비율

데이터 이동형(Type)	규모(CFSU)	비율
Entry(E)	144	26%
Exit(X)	49	9%
Read(R)	194	35%
Write(W)	170	30%
Total	554	100%

4. 기능점수와 COSMIC-FFP 사이의 변환 모델

4.1 기능점수와 COSMIC-FFP 사의 변환모델 관련 연구

소프트웨어 규모 측정은 소프트웨어 프로젝트의 적절한 관리를 위한 가장 중요한 것 중의 하나이다. 사실 개발될 소프트웨어의 규모를 안다는 것은 사람, 원료, 시간 그리고 얼마나 많은 비용이 들 것인지 등 필요한 자원을 결정하는 것이 가능해 진다. 따라서, 성공적인 소프트웨어 프로젝트 추진을 위해 개발 초기에 정확한 소프트웨어 규모를 산정하는 것은 매우 중요하며, 소프트웨어 규모 산정 방식의 최근 동향은 사용자가치 중심의 기능점수기법(FPA)을 선호하고 있다. 기능점수기법은 개발 초기단계에 적은 노력으로 소프트웨어 규모를 예측할 수 있다는 장점은 있으나, 지나치게 단순화시킨 기능점수 예측 방식으로 인해 실제 산출결과와 예측된 기능점수 사이에 측정 오차가 발생되며, 경영정보시스템에 한정되어 있어 실시간이나, 임베디드, 공학계산 등의 소프트웨어 규모 측정에는 적용이 불가능 하다. 이런 기능점수의 문제점을 보완하기 위해 COSMIC이 설립되고, 기능점수기법을 실시간 소프트웨어와 내장형 소프트웨어로 확장한 것이 완전기능점수 또는 COSMIC-FFP라 한다[40]. 따라서, 소프트웨어 규모 예측 기법 중 기능점수와 COSMIC-FFP를 3장에서 소개 하였으며, 본장에서는 기능점수와 COSMIC-FFP 사이의 변환에 대해 살펴보고, 3개의 실험적 연구를 연대순에 따라 소개한다.

3가지 연구 모두가 기능점수(I)를 독립변수로 COSMIC-FFP(C)를 종속변수로 하여 비선형 최소제곱법에 의해 보정된 1차 방정식을 사용하여 변환모델을 보여 주고 있다. 방정식은 식(22)와 같다.

$$C = a + b*I \quad (22)$$

첫째, Vogelezang과 Lesterhuis[41]는 [표 31]과 같이 11개의 데이터를 가지고 기능점수와 COSMIC-FFP 사이의 변환을 위한 1차 방정식을 처음으로 제시하였다. 방정식은 식(23)과 같다. T_I = 기능점수의 총합을 표시하며, T_c = COSMIC-FFP값의 총합을 표시한다. 이때 결정계수 R^2 = 0.99이다. 총 변동 중에서 회귀직선(E)에 의해 설명되는 변동 비율을 결정계수($0 \leq R^2 \leq 1$)라 하며, 값이 클수록 신뢰성 있는 회귀직선인 모델이 얻어진다.

$$T_c = -86.8 + 1.2 * T_i \quad (23)$$

[표 31] Vogelezang와 Lesterhuis 데이터[41]

ID	T_I	T_c
1	39	23
2	52	29
3	120	115
4	170	109
5	218	181
6	224	182
7	249	173
8	260	81
9	380	368
10	766	810
11	1424	1662

Fetcke [42]는 [표 32]와 같이 5개의 어플리케이션에 대해 IFPUG 4.1과 COSMIC-FFP 2.2 방식에 따라 기능점수와 COSMIC-FFP 값을 측정하였으나, 새로운 모델은 제안하지 못하였고, Vogelezang & Lesterhuis는 Fetcke이 측정한 데이터를 이용하여 식(24)를 제안하였다. 이 경우 R^2 = 0.98이다.

$$T_c = -6.2 + 1.1 * T_i \quad (24)$$

[표 32] Fetcke 데이터 [42]

ID	ILE	EI	EO	EQ	T_I	E	X	W	R	T_c
1	21	30	13	13	77	48	25	10	28	81
2	14	16	4	6	40					38
3	14	19	9	7	49					51
4	19	21	9	7	56					52
5	12	9	4	6	31					29

둘째, Abran, Desharnais and Aziz [43]은 [표 33]에 나타난 6개의 데이터를 이용하여 IFPUG 4.1와 COSMIC-FFP 2.2를 비교하였으며, 식(25)와 같은 모델을 제안했다. 이때 R^2 = 0.91이다.

$$T_c = 18 + 0.84 * T_i \quad (25)$$

[표 33] Abran 데이터[44]

ID	T_I	T_c
1	103	75
2	362	209
3	124	170
4	263	203
5	1146	934
6	570	675

마지막으로, Desharnais, Abran and Cuadrado-Gallego[45]는 [표 34]에 나타난 14개의 데이터를 이용해 IFPUG 4.1 와 COSMIC-FFP 2.2를 비교하였으며, 식(26)과 같은 모델을 제안했다. 이 경우 R^2= 0.93이다.

$$T_c = -3.11 + T_i \quad (26)$$

[표 34] Desharnais 데이터 [45]

ID	ILE	EI	EO	EQ	T_I	E	X	W	R	T_c
1	112	31	145	95	383	63	155	26	120	364
2	217	98	162	168	647	96	233	45	91	565
3	98	104	127	71	400	59	125	68	146	398
4	61	64	55	25	205	39	66	28	55	188
5	77	94	135	66	372	52	158	65	173	448
6	53	22	29	22	126	20	37	7	24	88
7	56	24	21	10	111	11	41	16	47	115
8	70	94	51	72	287	45	103	45	104	298
9	96	202	54	148	500	78	110	193	198	579
10	105	83	128	28	344	54	114	31	92	291
11	105	55	88	69	317	49	119	28	98	294
12	49	103	49	57	258	50	86	38	78	252
13	26	42	35	10	113	19	23	33	39	114
14	105	157	115	70	447	67	149	84	167	467

4.2 기능점수와 COSMIC-FFP 사의 변환 모델 개발

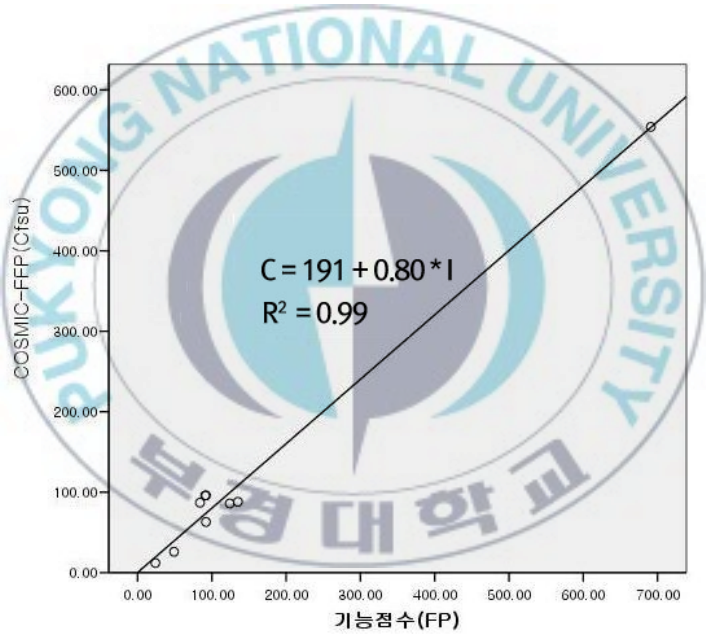
[그림 8]은 사례연구를 통해 COSMIC-FFP 3.0에 따라 직접 산출한 COSMIC-FFP 값을 종속변수로, IFPUG 4.1에 따라 측정한 기능점수를 독립변수로 두어 선형회귀 분석 결과를 그래프로 나타내고 있으며, [표 35]는 서브시스템별로 산출된 COSMIC-FFP 값과 기능점수를 나타내고 있다. 여기서, 기능점수 측정치는 I, COSMIC-FFP 측정치는 C로 표시하였으며, R^2는 결정계수를 나타낸다.

[표 35] 기능점수와 COSMIC-FFP 비교

서브 시스템명	I	C
예산관리 시스템	92	96
수입관리 시스템	49	26
계약관리 시스템	92	63
회계관리 시스템	24	12
인사관리 시스템	84	87
급여관리 시스템	124	86
총무관리 시스템	135	88
연구실적관리 시스템	91	96
합계	691	554

[표 36] 기능점수와 COSMIC-FFP 사이의 변환모델

모델	결정계수(R^2)
C= 191 + 0.80 * I	0.99



[그림 8] 기능점수와 COSMIC-FFP 사이의 변환모델

추정된 개발노력(시간)에 대해 단위시간 당 평균 소요 비용을 곱하면 개발 비용이 산출된다. 따라서 개발 노력이 산출되면 소프트웨어 개발에 투입될 비용의 예측이 가능해 진다. 현업에서도 Man Month보다는 기능점수를 통해 소프트웨어 개발비용을 산정하는 곳의 비중이 증가되고 있으며, 한국소

프트웨어산업협회에서 제공되는 소프트웨어산정대가 기준으로 기능점수가 제시 되었으며, 사회적 흐름에 따라 실시간 소프트웨어나 임베디드 소프트웨어의 비중이 증가되고 있는 추세이다. 하지만 기능점수기법을 통해서는 이런 소프트웨어의 규모를 추정할 수 없으며, COSMIC-FFP를 통해 가능하다. 그러나 소프트웨어산정대가의 기준은 기능점수로 되어 있어 추정된 COSMIC-FFP 규모를 통해 개발비용을 산정할 수 없다. 따라서 본 논문에서 제안한 기능점수와 COSMIC-FFP의 변환 모델을 이용하면 추정된 COSMIC-FFP값을 기능점수로 변환하여 소프트웨어산정대가에서 제시하는 표준 개발비용으로 추정할 수 있으며, 또한 소프트웨어 개발현업에서 바로 적용이 가능할 것이다.

5. COSMIC-FFP을 기반으로 한 개발 노력 추정 모델

5.1 COSMIC-FFP기반 개발노력 추정 모델 개발

소프트웨어 개발 시 중요하게 제기되는 문제점 중에 하나는 개발 초기 단계와 관련된 노력과 비용을 정확하게 산정할 수 있는 기업의 능력이다. 정확하지 못한 개발노력의 추정은 기업의 자원 낭비와 프로젝트의 비효율성을 불러일으키게 되고, 결과적으로 프로젝트의 실패를 가져오게 된다. 이런 개발노력 추정에 대한 중요성 때문에 소프트웨어 개발노력 추정 분야는 과거 30여 년 동안 활발한 연구가 수행되었으나, 소프트웨어 개발 노력과 비용에 영향을 미치는 요인과 그들 간의 관계가 아직 불명확한 관계로 구체적인 노력과 비용 추정 모델이 없는 실정이다[46].

따라서, 본장에서는 사례연구를 통해 직접 산출한 완전기능점수와 실제로 소프트웨어 개발에 투입된 개발노력 사이의 회귀분석을 통해 개발노력 추정 모델을 제안한다.

프로젝트를 수행하면서 소요되는 시간을 서브시스템별, 공정 단계별로 기록하여 [표 37]과 같은 수치를 얻었다. 총 개발시간은 3,035시간이며, 그 중 분석에 861시간, 설계에 1,129시간, 구현에 1,045시간이 투입 되었다. 개발시간의 측정은 실제 업무에 투입된 시간만을 기록 하였으며, 업무시간 중 휴식 시간은 카운트에서 제외하였다. 공정과정 중 시험에 대한 투입시간은 프로젝트 일정에서 제외시켜 별도로 카운트 하지 않았다. [표 38]은 통합정보시스템의 8개의 서브시스템별로 산출된 완전기능점수와 실제 투입된 개발시간을 제시하고 있다.

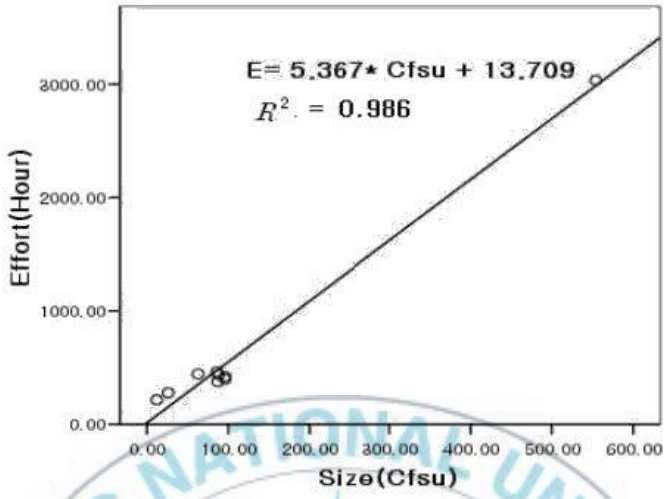
[표 37] 서브시스템별/공정별 투입 시간 (단위 : 노력(MH))

서브시스템명	총투입시간	분석	설계	구현	시험
예산관리	399	112	156	131	0
수입관리	279	80	96	103	0
계약관리	444	128	176	140	0
회계관리	216	64	78	74	0
인사관리	376	112	144	120	0
급여관리	462	128	176	158	0
총무관리	439	117	152	170	0
연구실적관리	420	120	151	149	0
합계	3,035	861	1,129	1,045	0

[표 38] 서브시스템별 COSMIC-FFP 및 실제개발시간

서브 시스템명	규모 (CFSU)	실제개발 투입시간 (E)
예산관리 시스템	96	399
수입관리 시스템	26	279
계약관리 시스템	63	444
회계관리 시스템	12	216
인사관리 시스템	87	376
급여관리 시스템	86	462
총무관리 시스템	88	439
연구실적관리 시스템	96	420
합계	554	3,035

[그림 9]는 사례연구를 통해 직접 산출한 CFSU값을 독립변수로, 프로젝트 개발에 실제 투입된 시간을 종속변수로 두어 선형회귀 분석 결과를 그래프로 나타내고 있으며, [표 39]는 COSMIC-FFP기반 개발노력 추정 모델을 제시 하고 있다.



[그림 9] 개발노력 추정 선형 회귀분석 모델

제안된 모델을 평가하기 위해 결정계수(Coefficient of determination, R^2)와 MMRE(Mean Magnitude of Relative Error)를 적용하였다. 주어진 데이터들의 변동을 적절히 표현할 수 있는 모델을 찾기 위해 결정계수를 이용한다. 종속변수의 값(E)은 독립변수의 값(CFSU)에 의해 결정되는 부분과 미지의 오차의 합으로 나타내며, 총 변동(주어진 데이터의 실제 개발노력 값) 중에서 회귀직선(E)에 의해 설명되는 변동 비율의 값이 클수록 신뢰성 있는 회귀직선인 모델이 얻어진다[47].

결정계수는 회귀분석의 모델 평가에 있어 다소 주관적인 평가내용이 반영되어지므로 MMRE를 확인하여 보았다.

$$RE = \frac{\text{추정치} - \text{실측치}}{\text{실측치}} \quad (27)$$

$$MRE(\text{Magnitude of RE}) = |RE| \quad (28)$$

$$\text{MMRE(Mean MRE)} = \frac{100}{n * \sum_{i=1}^{n} MRE} \qquad (29)$$

식(29)에서 100을 곱한 것은 MMRE를 백분율로 표시하기 위한 것이며, MMRE가 작은 값이면 모델은 평균적으로 좋은 모델임을 알 수 있다. Cont et al.[49]는 MMRE ≤ 0.25(25%)이면 개발노력을 예측하는 모델로 채택 가능한 것으로 고려하였다.

[표 39] COSMIC-FFP기반 개발노력 추정 모델과 성능

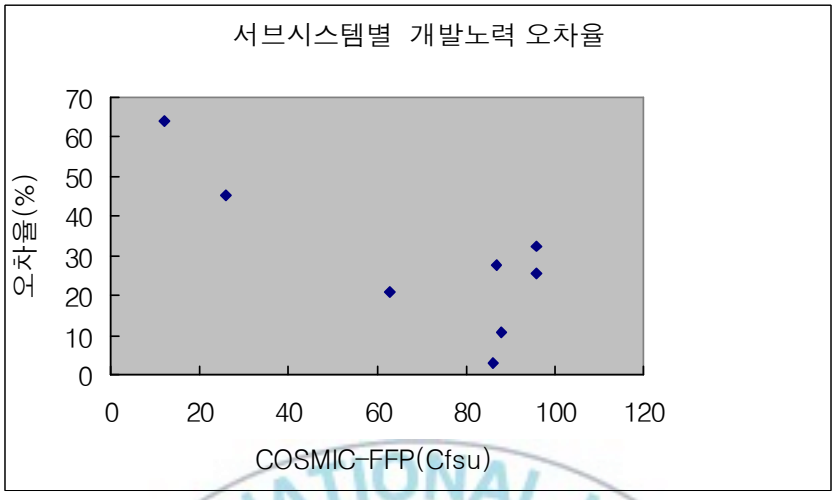
모델	결정계수(R^2)	MMRE
E= 5.367 · CFSU + 13.709	0.986	20.88

제안된 모델의 분석 결과 결정계수는 0.986으로 신뢰성 있는 값을 나타냈으며, MMRE은 약 21%로 Cont et al.[48]이 좋은 모델로 제시한 25% 보다 적은 값을 나타내고 있다. 따라서 본 논문에서 제시한 개발노력 추정 모델은 소프트웨어 개발노력을 예측하기에 상당히 정확성 있는 모델로 판명되었다.

5.2 시스템 복잡도를 이용한 개발 노력 가중치 계산 모델

가. COSMIC-FFP의 문제점

완전기능점수 기법은 데이터 처리, 실시간 시스템과 알고리즘 소프트웨어 등 광범위한 분야에 적용되는 장점을 갖고 있는 반면에 규모를 추정하는데 필요한 기능 요소들에 대한 가중치를 부여하지 않는 단점도 갖고 있다[49]. 사례연구를 통해 본 논문에서 제안한 모델에 의해 추정된 개발노력과 실제 투입된 개발노력 사이의 오차시간과 오차율을 계산하여 [표 40]에 제시하였다. 그리고, [그림10]은 실제개발노력과 모델 추정개발노력 사이의 오차를 분자로, 실제 투입된 개발 노력을 분모로 하여 백분율로 나타낸 것이다. 본 결과에서 알 수 있듯이 전체 24.9%의 상당히 높은 오차율을 보이고 있다. 이는 완전기능점수 기법의 단점에서 언급된 바와 같이 이를 보완해 줄 수 있는 가중치가 필요함이 증명된 것이다. 따라서 본 논문에서는 다음절에서 시스템 복잡도를 가중치로 적용할 수 있는 모델을 제안한다.



[그림 10] 서브시스템별 개발노력 오차율

[표 40] 모델을 적용한 추정 개발시간

서브 시스템명	규모	실제개발 투입시간	선형모델적용 추정개발시간	개발시간 오차 (절대치)	오차율
	(CFSU)	(hour)	(hour)	(hour)	(%)
예산관리 시스템	96	399	528	129	32.3
수입관리 시스템	26	279	153	126	45.1
계약관리 시스템	63	444	351	93	21
회계관리 시스템	12	216	78	138	63.9
인사관리 시스템	87	376	480	104	27.7
급여관리 시스템	86	462	475	13	2.8
총무관리 시스템	88	439	486	47	10.7
연구실적관리 시스템	96	420	528	108	25.7
합계	458	2,615	2,987	650	24.9 (평균오차율)

나. COSMIC-FFP 기반의 복잡도 계산 모델 개발

소프트웨어 기능을 기반으로 규모를 측정하는 방법들은 대부분 기능을 측정하는 과정에서 복잡도를 함께 고려하고 있다. 소프트웨어의 복잡도는 시스템 복잡도와 컴포넌트 복잡도로 구분할 수 있다. 하지만, COSMIC-FFP 기법은 소프트웨어 기능 프로세스의 컴포넌트 관점을 기술하고 있으나, 규모 추정과 관련된 기능요소들에 가중치를 부여하고 있지 않다[50]. 따라서 본 장에서는 COSMIC-FFP 기반 개발노력 추정 모델의 정확성을 높이기 위해 시스템 복잡도를 추정할 수 있는 모델을 제안한다. 시스템 복잡도는 추정된 개발노력 수치에 가중치로 적용함으로써 시스템 개발 초기단계에서 보다 정확한 개발노력을 예측할 수 있게 된다.

시스템 복잡도는 4가지의 데이터 이동 타입(들어가기, 나가기, 읽기, 쓰기)을 기반으로 측정 되어진다. 각 레이어에서 데이터 이동은 COSMIC-FFP의 규칙과 원칙을 적용하여 식별하였다.

시스템 복잡도 (SC_x)는 내부 복잡도($intraC_x$)와 외부 복잡도($interC_x$)로 구성된다. 내부 복잡도는 같은 레이어에서 콤포넌트 사이의 복잡도를 나타내며, 외부복잡도는 레이어 사이의 복잡도를 의미한다. 따라서, 본 논문에서는 G.Zayaraz, and P. Thambidurai[50]가 제안한 시스템 복잡도 식(30), (31), (32)를 이용하여 본 논문 4장에서 제안된 개발노력 추정 모델에 대한 가중치로 사용하였다. 내부 복잡도, 외부 복잡도 모두 범위는 0에서 1사이의 값을 가진다.

[표 41] 시스템 복잡도에 사용되는 매개변수

매개변수	심볼
Entry	E
Exit	X
Read	R
Write	W
Number of components	N
Layer	L

$$intraC_x = \sum_{i=1}^{L} \frac{(E_i * X_i)^2}{(N_i^2 * (N_i * (N_i - 1)))^2} \quad (30)$$

$$interC_x = \sum_{i=1}^{L-1} \frac{(E_{(i,i+1)} * X_{(i,i+1)})^2 + (R_{(i,i+1)} * W_{(i,i+1)})^2}{(N_i * N_{i+1})^4} \quad (31)$$

시스템 복잡도 = 내부 복잡도 + 외부 복잡도 (32)

위에서 제안된 식(30), (31), (32)을 사례연구에 적용하여 시스템 복잡도를 계산한 결과 [표 42]와 같은 값을 도출하였다. 데이터 이동 수는 CFFP V3.0을 기반으로 측정하였으며, 컴포넌트 수는 각 서브 시스템에서 기능별로 그룹을 구성하여 하나의 모듈(컴포넌트)로 간주하였다.

[표 42] 서브시스템별 시스템 복잡도

서브 시스템명	데이터 이동 수				컴포넌트 수	규모 (CFSU)	시스템 복잡도
	E	X	R	W			
예산관리 시스템	20	7	55	14	4	96	0.53
수입관리 시스템	8	5	10	3	3	26	0.54
계약관리 시스템	22	14	10	17	5	63	0.37
회계관리 시스템	6	1	4	1	2	12	0.56
인사관리 시스템	15	9	11	52	4	87	0.49
급여관리 시스템	28	9	22	27	6	86	0.05
총무관리 시스템	22	5	23	36	4	88	0.32
연구실적관리 시스템	23	6	54	16	4	96	0.52

다. 시스템 복잡도를 기반으로 한 개발노력 가중치 계산 모델 개발

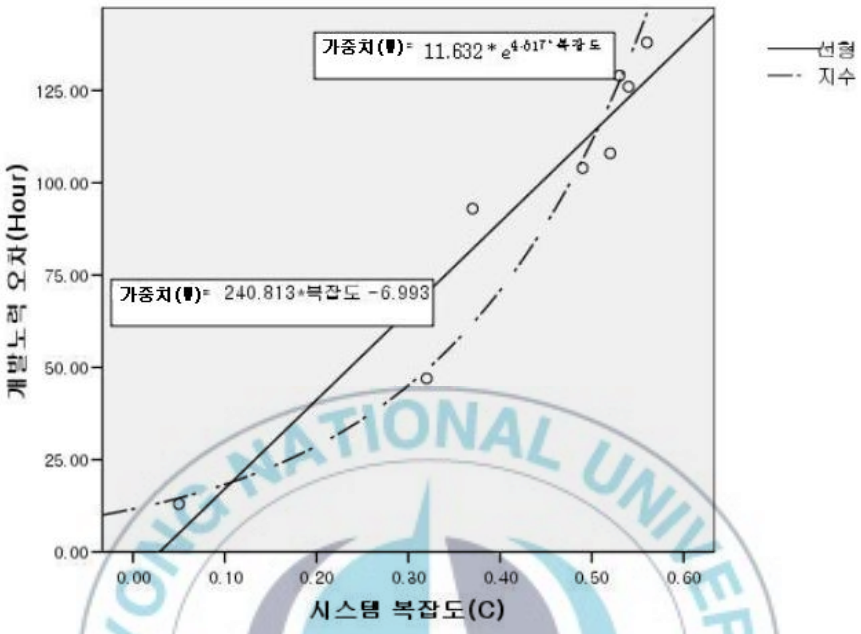
본 절에서는 시스템 복잡도(독립변수)와 실제개발 투입시간과 본 논문에서 제안된 개발노력 추정모델에 의해 산출된 개발노력 예측치 사이의 차이값(종속변수)에 대한 회귀분석을 통해 개발노력 오차를 보정할 수 있는 추정모델을 제안한다. 개발노력의 오차를 추정함으로써 COSMIC-FFP기반 개발노력 추정모델에 의해 추정된 개발 노력에 오차를 보정해 줌으로써 보다

정확한 소프트웨어 개발노력이 예측 가능하다. [표 43]는 서브시스템별 시스템 복잡도와 본 논문에서 제안된 COSMIC-FFP기반 개발노력 추정모델에 의해 산정되어진 개발노력 추정치와 실제 투입된 개발노력 사이의 오차를 나타내고 있다.

[표 43] 서브시스템별 시스템 복잡도와 개발노력 오차

서브 시스템명	복잡도 (독립변수)	개발노력 오차 (종속변수, 절대치)
예산관리 시스템	0.53	129
수입관리 시스템	0.54	126
계약관리 시스템	0.37	93
회계관리 시스템	0.56	138
인사관리 시스템	0.49	104
급여관리 시스템	0.05	13
총무관리 시스템	0.32	47
연구실적관리 시스템	0.52	108

[표 43]의 8개 서브시스템을 대상으로 시스템 복잡도와 개발노력 오차 사이의 회귀분석 결과는 [그림 11]에 제시하였다. [그림 11]에 나타난 시스템 복잡도를 이용한 개발노력 오차 추정 모델의 성능 분석 결과는 [표 44]에 제시되어 있다.



[그림 11] 개발노력 오차 추정 회귀분석 모델

[표 43]에서 제시된 데이터를 이용하여 COSMIC-FFP기반으로 산출된 시스템 복잡도를 독립변수로, 실제개발 투입시간과 본 논문에서 제안된 개발노력 추정모델에 의해 산출된 개발노력 예측치 사이의 차이값인 개발노력 오차를 종속변수로 두어 회귀분석을 통해 선형회귀 모델과 지수개발노력 회귀모델을 [표 44]과 같이 구하였다. 본 논문에서 제안한 개발노력 추정모델에 의해 산정된 개발노력 추정치에 시스템 복잡도를 적용하여 추정한 개발노력 오차를 보정하면 실제 프로젝트에 투입될 개발노력을 보다 정확히 예측할 수 있음으로 본 논문 5.2절에서 제안된 모델에 의해 산출되는 값을 가중치(W)라고 정의하였다. 두 모델의 성능평가 결과 지수 모델이 선형 모델에 비해 결정계수 및 MMRE 모두에서 좋은 것으로 판명되었다. 따라서 본 논문에서는 지수회귀 모델을 시스템 복잡도를 적용하여 개발노력 추

정치를 보정하기 위한 가중치 추정 모델로 선정하였다.

[표 44] 시스템복잡도를 이용한 개발노력 가중치 계산 모델

구분	모델	결정계수 (R^2)	MMRE
선형	가중치(W)= 240.813 · 복잡도 -6.993	0.921	6.64
지수	가중치(W)= $11.632 * e^{4.517 * \text{복잡도}}$	0.955	5.28

[표 43]에 나타난 8개의 서브시스템별로 시스템 복잡도를 적용하여 실제 소프트웨어 개발노력이 투입된 시간과 제안 모델에 의한 추정치 사이의 오차를 비교해 본 결과 CFSU 값이 85이상일 경우는 제안 모델에 적용한 개발노력 추정치가 실제 투입되는 개발시간보다 더 많이 투입될 것으로 예측되었으며, 85미만인 경우 추정치가 실제투입시간보다 적은 값을 나타냈다. 따라서 시스템 복잡도를 적용하여 모델 추정치에 대한 개발노력을 보정할 경우 CFSU값이 85이상인 경우 오차 가중치를 개발노력 추정치에서 마이너스하고, 85미만인 경우 반대로 오차 가중치를 합산해 주면 보다 정확한 개발노력을 예측할 수 있다.

5.3 시스템 복잡도를 적용한 COSMIC-FFP기반 개발노력 추정 과정 개발

제안된 모델을 통해 투입될 개발노력 추정치를 산정하고 여기에 시스템 복잡도를 적용한 가중치를 부여함으로써 보다 정확한 개발노력 예측이 가능

하다. 따라서, 다음과 순서로 개발노력을 추정하면 소프트웨어 개발 초기단계에서 보다 정확한 결과를 얻을 수 있다.

단계 1 : 본 논문에서 제안한 COSMIC-FFP 기반의 개발노력 추정 모델 E= 5.367 · CFSU + 13.709 을 통해 개발노력을 추정한다.

단계 2 : $$intraC_x = \sum_{i=1}^{L} \frac{(E_i * X_i)^2}{(N_i^2 * (N_i * (N_i - 1)))^2} \quad (33)$$

$$interC_x = \sum_{i=1}^{L-1} \frac{(E_{(i,i+1)} * X_{(i,i+1)})^2 + (R_{(i,i+1)} * W_{(i,i+1)})^2}{(N_i * N_{i+1})^4} \quad (34)$$

식(33)을 통해 시스템 내부 복잡도, 식(34)를 통해 외부 복잡도를 산출한다.

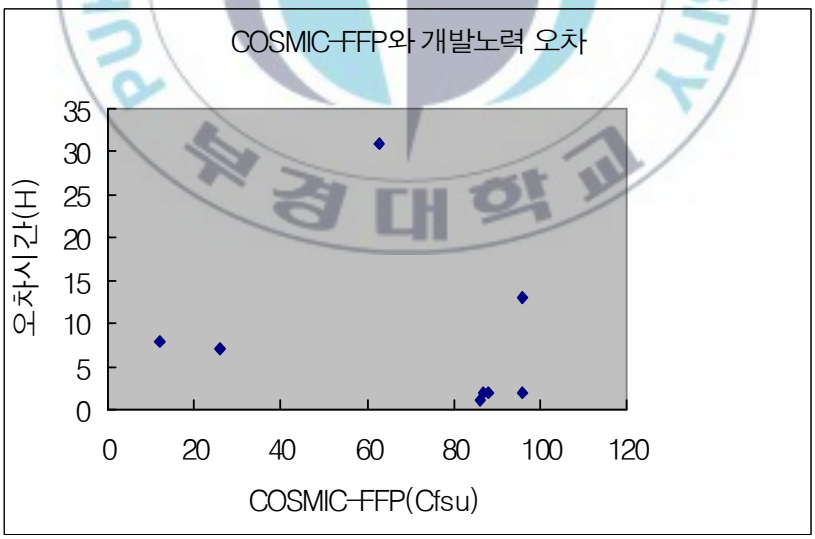
단계 3 : 시스템 복잡도 = 내부 복잡도 + 외부 복잡도 (35)
식(33), (34), (35)를 이용하여 시스템 복잡도를 산출한다.

단계 4 : 본 논문에서 제안한 개선된 복잡도를 이용한 개발노력 보정 모델

가중치(W)= $11.632 * e^{4.517 * 복잡도}$ 을 이용하여 개발노력 보정치를 가중치로 산출한다.

단계 5 : CFSU >= 85 이면 개발노력 추정치(E) - 가중치(W) (36)
CFSU < 85 이면 개발노력 추정치(E) + 가중치(W) (37)
추정된 CFSU 값에 따라 식(36) 또는 식(37)을 적용하여 보정된 최종 개발노력 추정치를 산출한다.

[표 45]는 사례연구를 통해 직접 측정한 개발노력과 본 논문에서 제안한 COSMIC-FFP기반의 개발노력 추정 모델을 통해 추정한 개발노력 사이의 오차를 나타내고 있다. 또한, [그림 12]는 서브시스템별 개발노력 추정치와 실제 투입된 개발노력 사이의 차이를 그래프로 표현한 것이다. [그림 12]와 [표 45]에 확인 할 수 있듯이 두 값 사이의 오차가 아주 적음을 알 수 있다. 이는 본 논문에서 제안한 COSMIC-FFP기반 개발노력 추정 모델의 정확도가 높음을 의미하며, 이를 통해 소프트웨어 개발현장에서 보다 정확한 개발노력 예측이 가능 할 것이다. [표 45]에 표시된 보정 후 개발노력 오차 값 중에서 계약관리시스템은 다른 7개의 서브시스템에 비해 오차가 상대적으로 큰 것을 확인할 수 있다. 그 원인은 계약관리시스템의 복잡도가 충분히 반영되지 못한 것으로 판단된다.



[그림 12] COSMIC-FFP 규모와 개발노력 오차

[표 45] 서브시스템별 개발노력 예측치 및 실제 개발시간

서브 시스템명	규모 (CFSU)	실제개발 투입시간 (hour)	보정치를 적용한 추정개발시간 (hour)	보정 전 개발노력 오차 (절대치, hour)	보정 후 개발노력 오차 (절대치, hour)
예산관리 시스템	96	399	401	129	2
수입관리 시스템	26	279	286	126	7
계약관리 시스템	63	444	413	93	31
회계관리 시스템	12	216	224	138	8
인사관리 시스템	87	376	374	104	2
급여관리 시스템	86	462	461	13	1
총무관리 시스템	88	439	437	47	2
연구실적관리 시스템	96	420	407	108	13

6. 결론 및 향후 연구방향

성공적인 프로젝트 수행을 위해서 개발초기 단계에서 소프트웨어 개발노력 등 자원을 예측하는 것은 매우 중요하다. 그리고 이는 소프트웨어 규모를 통해 가능하며 소프트웨어 규모를 어떻게 정량화 할 것인가에 대한 많은 연구가 진행되고 있다. 최근 소프트웨어 규모 산정 방식의 동향은 사용자 가치 중심의 산정방식인 기능점수 분석기법을 보다 선호하고 있다. 기능점수기법은 사용자관점에 기반을 두고 기능성으로 소프트웨어 규모를 정량화 하는 기법이다. 하지만, 기능점수 분석기법은 MIS영역 이외의 분야에서는 적용하기가 어렵다는 단점이 있다. 이런 기능점수의 문제점을 보완하기 위해 COSMIC이 설립되고, 기능점수기법을 실시간 소프트웨어와 임베디드 소프트웨어로 확장한 것이 COSMIC-FFP이다. 사회적 흐름에 따라 실시간 소프트웨어나 임베디드 소프트웨어의 비중이 증가되고 있는 추세이며, 이는 COSMIC-FFP기법을 통해 소프트웨어 규모 추정이 가능하다. 하지만, COSMIC-FFP로 측정된 소프트웨어 규모에 대한 개발노력을 추정할 수 있는 모델 연구는 미흡하다. 따라서 기존의 소프트웨어 규모 측정 방법에 기반하여 개발노력을 추정하기 위한 모델에 대한 연구와 함께 사례연구 결과에 따라 본 논문에서는 완전기능점수 기반의 개발노력 추정 모델을 제안하였고, 제안된 모델의 정확도를 높이기 위해 COSMIC-FFP의 개발단계에 적용되는 4가지 요소인 읽기, 쓰기, 입력, 출력을 이용한 시스템 복잡도를 추정하는 모델을 제안하였다. 그리고, 시스템 복잡도를 기반으로 가중치를 산출하여 추정된 개발노력을 보정할 수 있는 모델 또한 제안하고 평가 하였다. 소프트웨어 규모 측정은 소프트웨어 프로젝트의 적절한 관리를 위한

가장 중요한 것 중의 하나이다. 사실 개발될 소프트웨어의 규모를 안다는 것은 사람, 원료, 시간 그리고 얼마나 많은 비용이 들 것인지 등 필요한 자원을 결정하는 것이 가능해진다. 그리고, 한국소프트웨어산업협회에서 제시한 소프트웨어사업 대가의 기준 또한 기능점수기법을 기반으로 표준 개발비용을 제시하고 있다. 하지만, 기능점수기법으로는 실시간 소프트웨어나 임베디드 소프트웨어 규모에 대한 추정이 불가능하다. 따라서 본 논문에서는 기능점수기법과 COSMIC-FFP기법 사이의 변환 모델을 제안함으로써 추정된 COSMIC-FFP값을 기능점수로 변환 할 수 있다. 제안된 모델을 통해 소프트웨어사업 대가의 기준에서 제시하는 표준 개발비용의 추정 범위를 실시간 소프트웨어와 임베디드 소프트웨어까지 확장 가능하다.

본 논문에서 제안된 모델들은 실제 프로젝트 수행을 통해 얻어진 결과를 기반하고 있음으로, 현재 프로젝트 개발현장에서 제안된 모델을 이용해 보다 정확한 개발기간, 개발비용, 개발노력의 예측이 가능할 것이다. 또한, 소프트웨어 개발 초기단계에 개발노력을 추정할 수 있는 모델을 제안함으로써, 추정된 개발노력을 통해 개발에 소요될 비용이나 개발기간까지도 예측이 가능하다.

소프트웨어 개발에 영향을 주는 요소로서 시스템 복잡도 뿐만 아니라, 개발자의 스킬, 개발경험, 개발환경 등 다양한 것들이 있다. 소프트웨어 개발노력 예측 시 이러한 요소들을 다 반영된다면, 보다 정확한 예측이 가능할 것이다. 따라서 소프트웨어 개발에 영향을 주는 다양한 요소의 발굴 및 이를 개발노력 예측에 반영 할 방법에 대한 연구가 수행될 것이다. 또한 완전기능점수는 최근 들어 국제 표준화가 되었으며, 현재는 개발현장에서 적합성이 검증되고 있는 단계로 실제적으로 많은 데이터가 존재하지 않는다. 따라서 추후 다양한 종류의 실험데이터 수집을 통해 범용성을 높일 수 있는 모델에 대한 연구가 수행될 것이다.

참 고 문 헌

[1] Albrecht. A. J and Gaffney. J. E., "Software Function, Source Line of Code and Development Effort Predction : A Software Science Validation", IEEE Trans. on Software Eng., Vol.SE-9, No. 6, pp. 639-648, 1983.

[2] Kamer. Chris F., and Benjamin S. Porter "Improving the Reliability of Function Point Measurement: An Empirical Study", IEEE Transations on Software Engineering, Vol. 18, No. 11, pp.1011-1024, 1992.

[3] Bootsma. F., "Applying Full Function Points to Drive Strategic Business Improvement with the Real-Time Software Environment", Annual IFPUG Conference, New Orleans, 1999.

[4] Matson. J. E., Barrett B. E. and Mellichamp. J. M., "Software Development Cost Estimation Using Function Points", IEEE Trans. on Software Eng., Vol.20, No.4, pp.275~287, 1994.

[5] Albrecht. A. J., "Measuring Applications Development Productivity", Proceedings of IBM Application Dev., Joint SHARE/GUIDE Symposium, Monterey, CA., pp.83~92, 1979.

[6] Albrecht. A. J., "Measuring Application Development Productivity In Programming Productivity : Issues for the Eighties", IEEE Computer Society Press, 1981.

[7] Albrecht. A. J. and Gaffney. J. E., "Software Function, Source Line of Code and Development Effort Prediction : A Software Science

Validation", IEEE Trans. on Software Eng., Vol. SE-9, No.6, pp.639~648, 1983.

[8] Kemerer. C. F., "An Empirical Validation of Software Cost Estimation Models", Communication ACM, Vol.30, No.5, pp.416~429, 1987.

[9] Kemerer. C. F., "Reliability of Functional Point Measurement-A Field Experiment", Communications of ACM, 1993.

[10] Low. G. C. and Jeffery. D. R., "Function Points in the Estimation and Evaluation of the Software Process", IEEE Trans. on Software Eng., Vol.16, pp.64~71, 1990.

[11] Monaka. N., Kakural. A., Bukhary. E. and Azuma. M., "A Complexity Weighted Functional Size Metric for Interactive Software", Advanced Institute for Science & Eng., Waseda University, 2002.

[12] Ribu. K., "Estimating Object-oriented Software Projects with Use Cases", University of Oslo Department of Informatics, Master of Science Thesis, 2001.

[13] Matson. J. E., Barrett. B. E. and Mellichamp. J. M., "Software Development Cost Estimation Using Function Points", IEEE Trans. on Software Eng., Vol.20, No.4, pp.275-287, 1994

[14] Albrecht. A. J., "Measureing Applications Development Productivity", Proceedings of IBM Application Dev., Joint SHARE/GUIDE Symposium, Monterey, CA., pp.83-92, 1979

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

[16] Abran. A., Symons. C. and Oligny. S., "A Overview of COSMIC-FFP Field Trial Results", ESCOM 2001, London, England, 2001.

[17] Möller. K. H. and Paulish. D. J., "Software Metrics-A Practitioner s Guide to Improved Product Development", Chapman & Hall Co., New York, 1993

[18] Matson. J. E., Barrett. B. E. and Mellichamp. J. M., "Software Development Cost Estimation Using Function Points", IEEE Trans. on Software Eng., Vol.20, No.4, pp.275-287, 1994.

[19] Jones. C., "Programming Productivity", New York, McGraw - Hill, 1986.

[20] Condori-Fernandez. N., Abrahao. S. and Pastor. O., "Towards a functional size measure for object-oriented systems from require-ments specifications", Quality Software, 2004. QSIC 2004. Proceedings. Fourth International Conference, pp.94-101, 2004.

[21] Bailey. J. W. and Basili. V. R., "A Meta-model for Software Development Resource Expendirures", in Proc. 5th Int. Conf. on Software Eng., pp.107-116, 1981.

[22] Walston. C. E. and Felix. C. P., "A Method of Programming Measurement and Estimation", IBM System Journal, Vol.16, No.1, pp.54-73, 1997.

[23] Boehm. B. W., "Software Engineering Economics", IEEE Trans. on Software Eng., Vol.10, No.1, pp.7-19, 1984.

[24] IFPUG, "Function Point Users Group Practices Manual, Release 4.1", International Function Point Users Group, 1999.

[25] Umbers. P. and Miles. G., "Resource estimation for Web applications", Software Metrics, 2004. Proceedings. 10th International Symposium, pp.370-381, 2004.

[26] David Garmus and David Herron, "Function Point Analysis", Addison Wesley, 2001.

[27] Bootsma. F., "Applying Full Function Points to Drive Strategic Business Improvement with the Real-Time Software Environment", Annual IFPUG Conference, New Orleans, 1999.

[28] Ju Seok Park, Ki Won Chong, "A FFP-based Model to Estimate Software Development Cost", Korea Information Processing Society, Vol.10-D, No.4, pp.1137-1144, 2003.

[29] Symons. C., "COSMIC-FFP Measurement Manual, Version 3.0 (The COSMIC Implementation Guide for ISO/IEC 19761:2003)", Common Software MeasurementInternational Consortium, 2003.

[30] ISO/IEC FDIS 19761, "Software Engineering COSMIC-FFP-A Functional Size Measurement Method", 2002.

[31] Bertoa. F. and Vallecillo. A., "Usability Metrics for Software Components", Proceedings of the 8th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering(QAOOSE 2004), Oslo, Norway, June 2004.

[32] Zayaraz. G., Thambidurai. P., Madhu Srinivasan and Paul Rodrigues, "Software Architectural Quality Assessment Through COSMIC FFP", Proceedings of the National Conference on Product Development with Mechatronic Systems for Global Quality, PMGQ 2005, May2-3, pp.211-216. 2005.

[33] Larman. C., "Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process", Prentice-Hall, 2002.

[34] Karner. G., "Metrics for Objectory", Diploma Thesis, University of Linkoping, Sweden, No. LITH-IDA-Ex-934421, 1993.

[35] Santillo. L., Conte. M. and Meli. R., "Early & Quick function point: sizing more with less", Software Metrics, 2005. 11th IEEE International Symposium , pp.6-7, 2005.

[36] Jones. C., "Applied Software Measurement - Assuring Productivity and Quality", Chapter 2, New York, McGraw - Hill, 1991.

[37] (財)日本電力中央研究所, "ファンクションポイント法ハンドブック", 電力中央研究所 報告, 2000.

[38] Oligny. S., Bourque. P., Abran. A., and Fournier. B., "Exploring the Relation Between Effort and Duration in Software Engineering Projects", World Computer Congress 2000, Beijing, China, pp.175-178, August, 2000.

[39] 지경부, "소프트웨어사업대가의 기준", 고시 제2010-52호, 2009.

[40] Cuadrado-Gallego. J. J., Rodríguez. D., Machado. F. and Abran. A., "Convertibility between IFPUG and COSMIC Functional Size Measurements", in Proceedings of International Conference Product Focused Software Development and Process Improvement (PROFES) International Conference, Lecture Notes in Computer Science (LNCS), pp.273-283, July 2007.

[41] Vogelezang. F. and Lesterhuis. A., "Applicability of COSMIC Full Function Points in an administrative environment: Experiences of

an early adopter", Proceedings of the 13th International Workshop on Software Measurement, Shaker-Verlag, Montréal, Canada, Sept.22-25, pp.232-243, 2003.

[42] Fetcke. T., "The warehouse software portfolio, a case study in functional size measurement", Technical report no. 1999-20, Département d'informatique, Université du Quebec à Montréal, Canada, 1999.

[43] Abran. A., Desharnais. J.-M. and Azziz. F., "Measurement Convertibility: From Function Points to COSMIC-FFP", Proceedings of the 15th International Workshop on Software Measurement, Shaker-Verlag, Montréal, Canada, Sept.12-14, pp.227-240, 2005.

[44] IFPUG, "Function points counting practices manual 2.0", International Function Points Users Group, 1988.

[45] Desharnais. J. M., Abran. A. and Cuadrado-Gallego. J. J., "Convertibility of Function Points to COSMIC: Identification and analysis of functional outliers", Proccedings of the 17th International Workshop on Software Measurement, Shaker-Verlag, Palma de Mallorca, Illes Balears, España, Nov.5-8, pp.130-146, 2007.

[46] Levesque. G. and Bevo. V., "Comparing COSMIC-FFP and SLIM Back-Firing Function Points Size Measurements", ICS-SEA, 2001.

[47] Diab. H., Frappier. M. and St. Denis. R., "Formalizing COSMIC-FFP using ROOM ", Computer Systems and Applications, ACS/IEEE International Conference on. 2001, pp.312-318, 2001.

[48] Conte. S. D., Dunsmore. H. E. and Shen. V. Y., "Software Engineering Metrics and Models", Menlo Park., CA, Benjamin

Cummings, 1986.

[49] Mohamad Kassab, Olga Ormandjieva, Maya Daneva and Alain Abran, "Non-Functional Requirements Size Measurement Method (NFSM) with COSMIC-FFP", IWSM-Mensura 2007, LNCS 4895, pp.168 - 182, 2008.

[50] Zayaraz. G. and Thambidurai. P., "Quantitative Measurement of Software Architectural Qualities through COSMIC FFP", India Conference, 2006 Annual IEEE, pp.1-4, 2006.