The Journal of Korean
Institute of CALS/EC
Vol 7, No. 2, June 2002

UML 기반 소프트웨어시스템의 규모측정을 위한 기능점수분석기법의 응용에 관한 연구

안계중*, 이남용*

A Case Study on Applying Function Point Analysis Technique to Measure the Size of Software Systems based on UML

GyeJung Ahn, NamYong Lee

Abstract

Over the past decade, numerous software managers and engineers have been concerned with measuring the size and complexity of software systems. Function point analysis technique is one of the most popular software sizing techniques. A reasonable software development plan through cost and time estimation should be a prerequisite for the successful project at the beginning stage of the project. It is generally known that software size measurement is useful for this kind of estimation and the function point analysis technique would be more effective than the others. However, it is difficult to apply the technique to object-oriented methodology widely used in the software industry. Thus, the purpose of this study is to present a case study on how to apply function point analysis technique to sizing of the software systems based on UML. The results of this study can be useful to managers and engineers.

Key words: Function point analysis, UML, software sizing

* 숭실대학교 대학원 컴퓨터학과

1. 서론

컴퓨팅의 초창기에는 전체 컴퓨터 기반 시스템의 소요비용 중 소프트웨어가 차지하는 비용은 극히 일부분 이였다. 하지만 오늘날 소프트웨어는 모든 컴퓨터 기반 시스템에서 비용과 자원을 가장 많이 차지하는 요소 중 하나가 되었다[Roger S.]. 때문에 소프트웨어 개발 프로젝트에 소요되는 비용과 노력의 예측은 프로젝트를 성공적으로 이끌기 위한 필수적 요소라고 할 수 있으며, 이러한 예측 활동은 프로젝트의 초반에 수행되어야 보다 효과적일 수 있다. 이것은 합당한 방법을 통해서 산출된 신뢰성 있는 예측 값은 보다 체계적이고, 합리적인 계획을 수립할 수 있게끔 한다 라는 원칙에 입각한 것이다.

이러한 점을 바탕으로 하여 소프트웨어 프로젝트의 계획은 인도될 시스템의 크기에 대한 예측 값을 강조하는데[Kemerer et al, 1992], 이는 특정 소프트웨어 기반 시스템이나 제품을 만드는데 얼마나 많은 비용과 노력, 자원, 그리고 시간이 소요되는지를 보다 정확히 결정하기 위함이다. 즉, 소프트웨어 프로젝트의 예측과 이러한 예측을 기반으로 한 계획 수립의 목적은 소프트웨어 프로젝트에 소요되는 자원, 비용 및 일정을 합리적으로 도출하는 것이다[Roger S.], [Norman et al, 1997]. 또한 이러한 소프트웨어 프로젝트의 예측은 소프트웨어의 크기와 복잡도가 증가할 수록 그 중요성이 더욱 강조되며 특히 이들 요소들 중 개발노력은 가장 중요한 요소이다[Uemura et al, 1999]. 이에 따라 그동안 소프트웨어 크기를 중요한 인자로 포함하는 상당한 수의 노력모델 들이 제안되었으며 그 중 대표적인 것이 LOC(lines of codes)이다. LOC는 크기지향 소프트웨어 척도로서 품질과 생산성 측정치들을 생산된 소프트웨어 크기를 고려하여 표준화(normalize) 시킴으로써 도출할 수 있다[Norman et al, 1997]. 하지만 LOC를 비롯한 기존의 크기지향 척도들은 소프트웨어 개발 프로젝트를 측정하는데 가장 좋은 방법으로 수용되지는 못했다[JON, 1986]. 이것은 기존의 많은 소프트웨어 예측모델 들이 LOC나 KLOC를 핵심 입력으로 사용하고 있는 반면, 프로그래밍 언어에 의존적이며, 설계는 잘되었지만 길이가 짧은 프로그램의 적용에 적절치 못하고 비 절차적 언어를 쉽게 수용할 수 없다는 한계점을 가지고 있기 때문이라고 할 수 있다[Kemerer et al, 1992].

하지만 앞서 언급한 예측의 중요성에 따라 사용자의 관점에서 소프트웨어 개발을 측정하는 표준적 방법으로 기능점수분석(Function Point Analysis, 이하 FPA)이 제안되었다[IFPUG, 2000]. 정보시스템의 가능적 크기를 측정하는 FPA는 구현을 위해 사용되는 특정 언어나 기술에 종속적이지 않다. 이러한 FPA의 특징은 논리적 기능단위를 사용하여 소프트웨어의 크기를 기능점수로 측정한다는 것이다. 다시 말하면 비즈니스 모델 또는 사용자 요구사항을 통해서 정보시스템의 기능적 크기를 측정하기 때문에, 이를 통해 측정된 크기는 프로그래밍 언어, 설계 기술 또는 개발 기법과는 상관없는 일정한 값을 나타내는 것이다. 또한 기능점수는 소프트웨어 개발 프로젝트의 계획 수립을 위해 사용되기 때문에 개발 프로세스의

초기에 이용 가능하다는 이점을 갖는다 [Uemura et al, 1999]. 그러나 기능점수분석 수행에 있어서 동일한 조직 내에서 같은 제품을 가지고 측정한 기능점수 값이 차이가 발생하는 등의 문제점을 나타낸다[Kitchenham, 1997], [Low et al, 1990]

이와는 별도로 소프트웨어 개발 측면에 있어서 많은 패러다임이 제안되었다. 제안된 여러 패러다임 중에서 객체지향 개발은 객체를 사용해서 실세계의 복잡한 문제들을 보다 효과적으로 해결하는 뛰어난 능력을 가졌다. smalltalk, c++, Java와 같은 객체지향 언어 같은 소프트웨어 제품 개발의 환경이 제공되었고, 이외에도 객체지향 개발을 위한 다양한 기술들이 제안되었다. 결과적으로 현재 객체지향 개발은 산업계에서 폭 넓게 사용 되고 있다[Uehara et al, 1999], [Fioravanti et al, 2001]. 하지만 이러한 객체지향 기술을 소프트웨어 개발 프로젝트에 사용하는 것은 소프트웨어 및 소프트웨어 개발 프로세스의 새로운 문제점을 야기시킨다 [Basili et al, 1996]. 즉, 객체지향 소프트웨어에 적용할 수 있는 소프트웨어 측정 방법이 아직까지 수립되지 않았다는 것이다.

본 논문에서는 UML(Unified Modeling Language)을 사용한 객체지향 소프트웨어 개발의 초기 단계에서 기능점수를 계산하는 방법을 제안하고자 한다. UML을 사용한 객체지향 소프트웨어 개발 프로세스 중 요구사항 분석 단계에서는 유스케이스 모델을 수립하는데, 이 유스케이스는 기능점수의 트랜잭션의 개념과 상당한 유사점을 가진다. 이러한 유사점에 기반 하여 지금까지 기능점수를 객체지향에 적용하는 여러 연구들이 수행되어 왔다. 하지만 기존의 연구들이 기능점수 또는 UML의 표준에서 정의한 개념들을 부적절하게 사용하거나 기능점수분석을 수행한 소프트웨어 개발 프로세스의 단계가 적합하지 않는 등의 문제를 드러내고 있다. 따라서 본 논문에서는 객체지향 개발, 특히 UML을 사용한 개발 프로세스 중 초기 단계에서 기능점수분석 수행 방법을 제시하는 것을 목적으로 하고, 이를 위해서 기존의 관련 연구를 분석하여 기능점수와 UML의 표준에 따라 미비점을 보완, 정제하는 작업을 수행한다.

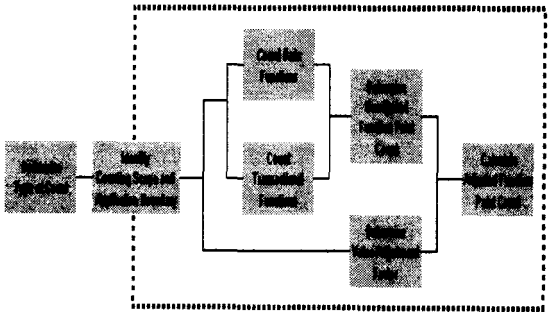
2. 관련연구

2.1 전통적 기능점수 분석

기능점수는 지난 70년대에 Allen Arbrecht에 의해 소프트웨어 프로젝트의 크기계산, 예측, 측정을 위한 방법으로 개발되었다 [sym91]. 기능점수는 사용자 관점으로 문제의 크기를 측정하며, 그 원칙은 개발에 소요되는 비용과 노력을 개발 프로세스의 초기에 예측하기 위해서 요구사항 명세에 초점을 두는 것이다. 또한 기능점수는 구현에 사용되는 기술에 독립적으로 소프트웨어의 크기를 측정할 수 있다. 이와 같은 이유로 기능점수는 상이한 기술들을 사용한 프로젝트들간의 성능비교를 가능하게 하며, 더불어 프로젝트 진행 초기에 요구되는 노력을 예측할 수 있게 한다.

기능점수의 계산 과정은 우선 기능점수의 유형을 결정한 후 계산 범위와 어플리케이션 경계를 식별한다. 그리고 외부 변수들

을 유형 별로 구분하여 각각 복잡도를 계산한 후에 미조정 기능 점수(unadjusted function point), 값 조정 인자, 조정된 기능점수(adjusted function point)를 계산한다. <그림 1>은 기능점수의 계산 절차를 나타낸다



<그림 1> 기능점수 계산과정

2.2 Karner의 UCP(Use Case Points)

소프트웨어 개발에 있어 객체지향 접근법은 주요한 흐름이며 UML은 개발하고자 하는 시스템을 모델링 하는데 적합하다. UML의 여러 모델 중 유스케이스는 시스템의 동적인 면을 모델링 하는데 사용된다[Basili et al, 1996]. 유스케이스 모델은 시스템의 기능적 요구사항을 나타내며 이는 유스케이스 모델이 개발 프로세스 초기에 요구사항 명세에 따라서 구축되 기 때문에 프로젝트 초기에 구축하고자 하는 시스템의 크기를 정의하는데 사용될 수 있게 한다.

1993년에 Gustav Karner는 객체지향방법을 통해 개발된 프로젝트의 예측과 크기 계산을 위한 "유스케이스 포인트(UCP: Use Case Point)"를 제안하였다. 본 방법은 기능점수분석과 MK II 기능점수분석의 확장으로서 이들 방법과 동일한 사상, 즉 소프트웨어 크기 측정의 기본은 사용자 관점에서의 기능성에 있다는 것에 기본을 두고 있다[Kirsten, 2001]. 그러나 이러한 유사성에도 불구하고 실제 본 측정방법은 기능점수분석에서 나타내는 기능적 크기에 기반하고 있지는 않다. 이러한 점 때문에 기능점수에 측정된 크기를 직접적으로 대응시키지 못한다[Fetcke et al, 1997].

실제 UCP는 시스템의 분석을 통해서 계산될 수 있으며 첫번째 단계는 식별된 액터와 유스케이스를 유형에 따라 분류하는 것이다. 분류된 액터와 유스케이스는 유형에 따라 가중인자가 할당되고 이를 계산하여 비조정 액터 가중치(UAW: unadjusted actor weights)와 비조정 유스케이스 가중치(UUCW: unadjusted use case weights)를 산출하고 이를 합산하여 비조정 유스케이스 점수(UUCP: unadjusted use case point)를 산출한다. 여기에 기술적 복잡 요소(Technical Complexity Factors)와 환경적 요소(Environmental Factors)를 이용하여 최종적으로 조정 유스케이스 점수(UPC)를 산출한다. 산출 공식은 다음과 같다.

UAW + UUCW = UUCP
TCF = 0.6 + (0.01 *Tfactor)
EF = 1.4 + (-0.03 * Efactor)
UPC = UUCP * TCF * EF

본 UCP의 문제점은 첫째로 유스케이스의 구조나 작성방법을 구체적으로 다룬 연구가 드물다는 것이다[Kirsten, 2001]. 때문에

유스케이스 모델과 명세의 상세도는 다양할 수 있다. 이러한 점은 유스케이스의 복잡도 측정을 어렵게 만들며 결과의 신빙성을 감소시킬 소지가 있다. 두 번째로 본 측정방법은 유스케이스 모델에서 나타나는 포함관계와 확장관계를 고려하지 않았다는 것이다. 또한 Gustav Karner가 제시한 유스케이스 점수 당 소요 노력 량의 인자들이 실제 프로젝트 적용 결과와 차이가 발생한다는 것이다. 한 예로 프로젝트에 참여 인력의 경험 수준을 나타내는 EF인자는 소프트웨어 개발 업체마다 상이할 수 있다. 따라서 동일한 요구사항을 통해 산출된 결과값이 업체마다 다르게 산출될 수 있다[서예영, 이남용, 2001].

2.3 Jacobson 방법론을 적용한 기능점수분석

Thomas Fetke는 기능점수분석에 객체지향 접근방법을 적용시킨 방법을 제안했다. 본 방법은 Object-oriented jacobson approach, 즉 OOSE를 기반으로 개발되어진 소프트웨어의 기능적 크기를 측정하기 위해서 측정 프로세스를 지원하는 간결한 규칙의 집합을 바탕으로 기능점수 모델에 유스케이스 모델을 직접적으로 적용시키는 방법을 제안했다.

본 방법은 국제 기능점수 사용자 그룹(IFPUG)의 측정 수행 지침(Counting Practices Manual)에서 정의된 표준 기능점수 분석을 객체지향 개발 프로세스의 분석 단계에서 개발된 모델에 적용하여 기능점수를 측정하는데 주안점을 두고 있다. 이러한 적용은 기능점수 분석이 구현에 사용되는 기술에 독립적일 수 있다는 것과 요구사항 모델은 사용자의 요구에 기반 한 기능적 요구사항 명세를 공식화할 수 있다는 것을 바탕으로 수행된다. 이것은 기능점수 분석이 객체지향 폐러 다임에 사용될 수 있다는 것을 증명하며, 따라서 다른 방법론으로 개발된 소프트웨어에 동일한 방법으로 수행한 측정 결과와 상호비교가 가능하다는 것을 나타낸다[Fetcke et al, 1998].

본 논문은 OOSE로 개발된 소프트웨어 어플리케이션에 기능점수를 위한 계산규칙을 적용하기 위하여, 기능점수와 OOSE의 개념과 용어의 유사성과 차이점을 분석하여 구체적인 사상(mapping) 규칙을 제시하였다. 하지만 Gustav Karner의 방법과 마찬가지로, 본 방법 또한 유스케이스의 구체적인 상세도를 제시하지 않고 있다. 또한 제시된 각 단계들 중 일부의 항목이 객체지향 개념에 대한 고려가 부족하다.

2.4 Sequence 모델을 이용한 기능점수분석

Uemura는 UML을 사용한 설계명세를 위한 비조정 기능점수의 계산에 주안점을 둔 상세 기능점수 측정 규칙에 관한 연구를 제시하였다[Uemura, 1999]. 이를 위해 Uemura는 UML에 기반 한 요구사항과 설계 명세에 IFPUG 버전의 다섯 단계를 적용시키고, 데이터 기능 유형을 계산하는 규칙과 트랜잭션 기능 유형을 계산하는 규칙을 UML의 개념에 입각하여 정의하였다. 이를 위해서 데이터 기능 유형을 계산하는 규칙을 정의할 때는 설계 단계에서 도출된 객체들을 이용하고, 트랜잭션 기능 유형을 계산하는 규칙을 정의할 때는 요구사항과 설계 명세 단

계에서 도출된 시퀀스 다이어그램(sequence diagram)을 이용하였다. 명세에 적용된 기능점수분석의 단계는 전통적 기능점수 분석의 단계와 동일하나 각 단계 마다 적용된 규칙은 객체지향의 특성에 맞게 변형되었다. 또한 국제 기능점수 사용자 그룹(IFPUG)의 규칙에 따라, 시퀀스 다이어그램 상에서 데이터 기능으로 식별된 특정 객체에 의해 교환되는 각 메시지를 트랜잭션 기능의 후보로 간주하였다.

본 연구에서 제안된 방법은 요구사항과 설계 단계에 적용하는 것을 기본으로 한다고 밝히고 있다. 하지만 본 방법은 객체지향 소프트웨어 개발 프로세스의 일반적 단계에 비추어 분석 단계를 누락시키고 있다. 또한 기술적인 고려가 아닌 사용자의 관점으로 드러난 기능을 식별하는 계산 범위 식별 단계에 있어 시퀀스 다이어그램에 나타난 객체의 유형에 의해서 계산 범위를 결정하고 있다. 그리고 개발 프로젝트에서 식별된 객체들이 여러 시퀀스 다이어그램에서 중복되어 나타나는 것을 고려하지 않고 있으며 객체들간의 특수한 관계, 즉 상속과 집합연관 등을 고려하지 못했다.

2.5 기존 연구의 평가

Gustav Karner가 제안한 방법의 문제점은 우선 유스케이스를 어떠한 방식으로 작성해야 하는지에 관한 기존 연구가 없다는 것이다[Kirsten, 2001]. 즉 이것이 의미하는 바는 유스케이스 모델의 추상화 정도나 유스케이스 명세의 상세도를 규정하지 못한다는 것이다. 이러한 점은 유스케이스의 측정을 어렵게 한다[John]. 또 다른 문제점은 유스케이스 모델의 포함, 확장 관계를 고려하지 않았다는 점이다.

Fetcke의 연구가 가지는 문제점은 Gustav Karner의 문제점과 유사하다. 즉 Fetcke가 제안한 각 단계에서는 요구되는 모델과 그 문서들의 상세도를 단순히 '비교적 높은 수준'이라고 명시하였을 뿐 추상화 정도와 상세도에 대한 제약이나 기준을 제시하지 않고 있다. 또 다른 문제점은 데이터 요소의 유형을 결정하고 이를 계산하는 단계의 제안은 객체지향개념과 전통적 기능점수 방법을 충실히 사상 시킨 반면 트랜잭션 요소의 유형 결정과 계산에 대해서는 객체지향개념의 고려가 거의 없으며 단순히 국제 기능점수 사용자 그룹(IFPUG)의 측정 수행 지침(Counting Practice Manual)의 참조만을 지시하고 있다. 이것은 Fetcke의 연구가 구축하고자 하는 시스템의 정적인 부분만을 고려했다는 것을 미루어 짐작할 수 있다.

마지막으로 Uemura의 연구가 가지는 문제점은 우선 연구의 적용범위를 객체지향 프로세스 중 요구사항과 설계 단계로 상정한 반면, 연구의 진행에 있어서는 유스케이스 모델에 대한 논의가 전혀 포함되어있지 않다. UML이 개발 프로세스에 독립적[Basili, 1996]이기는 하지만 UML을 적용한 객체지향 프로젝트 중 가장 중요하게 여겨지는 유스케이스 모델을 누락시킨 것은 논의가 여지가 있다고 볼 수 있다. 또한 Uemura가 제안한 단계 중 데이터 요소의 유형을 결정하고 계산하는 과정이 지나치게 단순화 되어 있다. 또한 트랜잭션 유형 결정을 위해 제안된 패턴이 전통적 기능점수 분석에서 요

구하는 사항을 충실히 이행하지 못한 문제점을 갖고 있다.

3. 제안 방법

본 논문에서 제시하는 모델은 기능점수분석의 표준인 국제 기능점수 사용자 그룹(IFPUG)의 측정 수행 지침(Counting Practice Manual)과 OMG에서 객체 지향 분석설계의 표준으로 제시한 통합 모델링 언어인 UML을 기반으로 하여 UML을 적용한 객체지향 개발에 적용 가능한 기능점수 계산 방법을 제안하고자 한다. 제시되는 모델의 각 단계는 국제 기능점수 사용자 그룹(IFPUG)의 측정 수행 지침(Counting Practice Manual)에서 정의된 기능점수분석 단계를 기본으로 한다. 그리고 각 단계에서 정의된 세부규칙과 개념을 이에 상응하는 객체지향 패러다임과 UML 개념으로 사상 시킨 후 계산을 수행한다. 이에 따라 제시되는 모델은 다음과 같은 특징을 가진다.

특징 1) 객체지향 소프트웨어 개발에 기능점수의 효과적인 적용을 위해서 각 모델의 개념 간 유사성과 차이점을 식별하고, 사상(mapping)의 적절성을 판단한다.

특징 2) 기능점수분석에 적용되는 UML의 구성요소 들의 상세화 정도 및 적용범위를 규정하여 측정 단계의 신뢰성을 높이고, 측정 결과의 신빙성을 높인다.

특징 3) 객체지향 소프트웨어 개발 프로세스와 기능점수분석의 적용 단계 간의 적절한 사상을 수립하여 적용을 용이롭게 한다.

3.1 제안 방법의 절차

단계1: 기능 점수 계산 유형의 결정

첫번째 단계는 국제 기능점수 사용자 그룹(IFPUG)의 측정 수행 지침(Counting Practice Manual)에 정의된 것과 같이 기능점수의 계산 유형을 결정하는 것이다. 기능점수의 계산은 프로젝트나 어플리케이션 모두에 적용될 수 있다. 마찬가지로 본 논문에서 제안하는 모델 또한 그 적용범위가 동일하지만 공통적으로 적용되는 요구사항은 기능점수 계산의 대상은 반드시 UML로 모델링 되어야 하며, 그에 따른 산출물이 작성되어 있어야 한다는 것이다.

단계2: 계산 범위와 어플리케이션 바운더리(boundary)의 식별

계산 범위는 실제 기능점수계산에 포함되는 기능성을 규정하는 것이며 어플리케이션 경계는 사용자와 측정되는 소프트웨어 간의 경계(border)를 의미한다. 이러한 계산 범위와 어플리케이션의 경계는 그 목적에 의해서 정해지며 그 목적은 사용자의 관점에 따라서 정해진다. 즉 계산 범위와 어플리케이션의 경계는 사용자의 관점에 따라서 어플리케이션과 외부 어플리케이션 또는 도메인 간의 구분을 나타낸다[Fetcke, 1998]. UML에 있어서 이러한 개념은 액터(actor)와 시스템 바운더리(system boundary)에 의해서 나타난다. 액터는 시스템 바깥에 위치하며 쓰임새와 교류할 때 유스케이스 사용자들이 맡고 있는 역할을 나타낸다. 시스템 바운더리는 유스케이스를 포함하는 시스템의 경계를 정의한다[Frank et al, 2001]. 특히 바운더리

를 구성하는 유스케이스 또한 사용자의 관점에서 요구되는 행동, 즉 기능성만을 나타낼 뿐, 그 행동을 수행하는 방법을 규정하지는 않는다.

기능점수의 경계가 기술적인 고려가 아닌 사용자의 관점에 기반 하여 결정된다는 특징은 앞서 언급한대로 UML의 액터, 유스케이스, 바운더리 등의 요소로 구성되는 유스케이스 모델과 개념적 유사성을 가진다. 따라서 본 단계의 수행은 유스케이스 모델의 구축을 통해서 완료될 수 있다. 하지만 개념적 유사성이 완벽한 동일성을 나타내는 것이 아님을 주목해야 한다.

우선 UML에서의 액터는 앞서 정의한 것과 같이 유스케이스와 교류하는 사용자의 역할을 나타내는 것으로써, 사람이나 하드웨어 장비, 또는 다른 시스템을 나타낸다 [Basili et al, 1996]. IFPUG의 사용자(user)의 정의는 소프트웨어와 상호작용 하거나 의사소통을 수행하는 사람이나 사물을 나타낸다 [IFPUG, 2000]. 두 표준이 나타내는 액터(actor)와 사용자(user)의 개념이 약간의 차이를 갖기 때문에 액터를 기능점수분석의 사용자에 직접 사상시키는 것은 무리가 있다고 판단된다.

따라서 본 논문에서는 기본적으로 유스케이스 다이어그램 상의 액터 중 유스케이스에 행동을 요청하는 것, 즉 연관관계의 시작점을 갖는 액터를 기능점수분석의 사용자와 동일하게 상정한다. 이것은 유스케이스에게 행동을 요청하는 능동적 역할을 수행하는 액터를 의미한다. 그 반대의 경우는 유스케이스가 수행하는 행위에 있어서 필요한 데이터를 요청하는 경우를 의미하며, 이런 경우의 액터는 사용자의 개념이 아닌 참조 데이터로서 간주한다. 또 UML의 시스템 바운더리(system boundary)의 개념은 기능점수분석의 바운더리의 개념과 큰 차이점을 가지지는 않는다. 하지만 사용자와 측정되는 소프트웨어 간의 경계만을 나타내는 기능점수의 바운더리와는 달리 UML의 시스템 바운더리는 유스케이스 또는 유스케이스 패키지(package)로 구성된다.

단계3: 데이터 기능의 계산

유형의 결정

본 단계는 시스템에서 사용되는 데이터 기능을 식별하고 이를 계산하는 것을 목적으로 한다. 우선 국제 기능점수 사용자 그룹(IFPUG)의 측정 수행 지침(Counting Practice Manual)은 데이터 기능이라는 것을 사용자에게 내외부 데이터의 요구사항을 충족시키기 위해 제공되는 기능성이라고 정의하고 있으며, 그 유형을 내부논리파일(ILF)과 외부논리파일(EIF)로 규정하고 있다. 여기서 사용되는 파일이라는 개념은 일반적인 데이터 처리 개념이나 데이터의 물리적 구현에서 언급되는 것이 아닌 논리적으로 연관성을 가지는 데이터의 그룹을 의미한다 [IFPUG, 2000]. UML에서 이러한 기능점수분석에서 사용되는 파일에 대응하는 것은 객체 혹은 클래스 라고 할 수 있다[Fetcke, 1998]. 하지만 데이터와 데이터를 조작하는 처리작업을 동시에 가지는 클래스의 특징 [Roger]과 추상화가 높아짐에 따라서 다양하게 나타나는 클래스의 유형 등을 고려할 때 클래스와 파일의 일대일 사상은 무리가 따

른다. 따라서 본 단계에서는 클래스 중 파일에 상응하는 것들을 고려하는 것을 제안한다.

이러한 것은 Fetke가 자신의 논문에서 사용한 Jacobson의 도메인 객체 모델로 식별된 클래스 중 엔티티 객체(entity object)와 같은 것이 있다. 또한 RUP의 비즈니스 모델링(Business Modeling) 단계에서 구축되는 비즈니스 오브젝트 모델 상의 비즈니스 엔티티가 본 요구에 부합한다고 할 수 있다. 또한 본 단계에서 사용할 수 있는 모델로 비즈니스 타입 모델이 있다. 비즈니스 타입 모델은 구축하고자 하는 시스템에 의해서 관리되어야 할 구체적인 비즈니스 정보를 포함하는 모델이다[Chessman2000].

위에 제시한 모델들을 통하여 후보 객체를 식별한 후에는 그 유형을 결정하여야 한다. 유형의 결정은 국제 기능점수 사용자 그룹(IFPUG)의 측정 수행 지침(Counting Practice Manual)에서 정의하고 있는 바를 따른다.

복잡도의 계산

식별된 객체들의 유형이 결정되었으면 객체들의 복잡도를 계산하여야 한다. 국제 기능점수 사용자 그룹(IFPUG)의 측정 수행 지침(Counting Practice Manual)은 복잡도 계산을 위해서 데이터 요소 타입(DET)와 레코드 요소 타입(RET)의 수를 식별할 것을 지시한다. 따라서 우리는 본 단계에서 DET와 RET, 그리고 제시된 세부규칙에 대응하는 UML의 요소들을 식별하여 계산을 수행한다.

데이터 요소 타입은 사용자가 인식 가능하고 유일하며, 반복되지 않는 필드나 속성을 정의하는 것이며 이것은 클래스의 속성에 대응한다고 할 수 있다[Uemura, 1999]. 그리고 레코드 요소 타입은 ILF나 EIF 내에 포함된 데이터 요소들로 사용자가 인식 가능한 서브 그룹을 정의하는 것으로서 식별된 후보 객체들과 연관관계를 맺고 있는 객체들에 대응한다. 이러한 대응관계에 기반한 계산 규칙은 다음과 같다.

a. 식별된 각 후보 클래스의 속성을 하나의 DET로 계산한다.
b. 클래스 간 1:n 관계 당 n쪽의 클래스에 하나의 DET를 추가한다.
c. 집합연관(aggregation) 혹은 복합연관(composition) 관계에 있는 객체의 수를 주 클래스에 각각 하나의 RET로 계산한다.
d. 집합연관(aggregation) 혹은 복합연관(composition) 관계에 있는 객체의 각 속성을 주 클래스에 하나의 DET로 추가한다.
e. 상속 관계에 있어서 자식 클래스가 부모 클래스 외의 클래스와 관계를 가지지 않는 경우 자식 클래스의 각 속성을 부모 클래스에 하나의 DET로 추가하고, 자식 클래스의 수를 부모 클래스에 RET로 계산한다.
f. 상속 관계에 있어서 부모 클래스가 추상 클래스인 경우 자식 클래스의 RET로 계산한다.
g. 별도의 관계를 가지지 않는 클래스는 하나의 RET로 계산한다.
h. 액터(actor) 중 기능점수분석의 사용자(user)로서 사상 되지 않은 것 중

유스케이스에 데이터를 제시하는 것을 각각 DET로 계산한다.

단계 4: 트랜잭션 기능의 계산

유형의 결정

본 단계는 시스템에서 사용되는 트랜잭션 기능을 식별하고 이를 계산하는 것을 목적으로 한다. 우선 국제 기능점수 사용자 그룹(IFPUG)의 측정 수행 지침(Counting Practice Manual)은 트랜잭션 기능을 어플리케이션이 사용자에게 데이터 처리를 위해 제공하는 것이라고 정의하고 있으며, 그 유형을 외부입력(EI)과 외부출력(EO) 그리고 외부질의(EQ)로 규정하고 있다. 여기서 사용되는 '외부'라는 개념은 단계 2에서 정의한 바운더리의 바깥쪽을 나타내며, 측정하고자 하는 시스템과 연관을 갖는 사용자(user)를 의미한다. 또한 트랜잭션의 세가지 유형 모두는 사용자에게 의미 있는 가장 작은 작업 단위로 독립적인 단위로 정의되어 있다.

UML에서 이러한 기능점수분석에서 사용되는 트랜잭션 기능에 대응하는 것은 바운더리에 포함되는 유스케이스 라고 할 수 있다. 유스케이스는 시스템 바깥에 있는 것과 시스템 간 교류를 나타내는 순차(sequence)들의 모임을 설명한다[Booch, 1998]. 즉 유스케이스는 하나의 순차가 아닌 여러 개의 순차들을 통해서 이루어진다. 여기서 사용되는 순차라는 개념은 하나 이상의 트랜잭션을 포함한다고 할 수 있다. 따라서 기능점수계산을 위해서는 각 유스케이스 내에서 트랜잭션을 식별해내야 한다. 이러한 작업을 위해 요구되는 문서 혹은 모델은 유스케이스의 행동을 정형적인 방법으로 명세한 유스케이스 명세서와 이것에 의거하여 작성된 시퀀스 다이어그램이다.

하지만 단 한번에 완벽한 유스케이스 모델을 만들어 낼 수는 없다[Armour, 2001]. 유스케이스 모델은 복잡한 실세계를 표현하며 반복적으로 개발된다. 유스케이스는 요구사항에 대하여 좀 더 많은 정보를 정제하는 과정을 거치게 되며, 이러한 요구사항 분석 과정이 진행됨에 따라 문제에 대하여 좀 더 많이 이해할 수 있게 된다. 이러한 이해는 명세의 상세도를 높이면서 다양한 추상화 정도를 나타내게 된다.

Frank Armour는 이러한 명세의 상세도에 따라 유스케이스 명세를 시스템의 개념적 설명을 제공하는 초기 유스케이스 명세(Initial use case description), 이를 좀 더 자세하게 서술한 기본 유스케이스 명세(Base use case description), 정련 유스케이스 모델(Elaborated use case description), 세가지 단계로 정의했다[Armour, 2001]. 기본 유스케이스 모델은 유스케이스의 경로와 이상적인 행위에 주안점을 두며, 예외 흐름이나 대안흐름은 기술하지 않는다. 본 논문에서는 명세의 상세도를 기본 유스케이스 명세에 맞추며, 추가적으로 예외흐름과 대안흐름의 발생 조건을 명시할 것을 제안한다.

다음으로는 제안된 요건사항을 충족시키는 유스케이스 명세와 이에 기반 하여 작성되어진 시퀀스 다이어그램을 분석하여 각 트랜잭션의 유형을 식별해내야 한다. 본 단계에 사용되는 시퀀스 다이어그램은 단계 2와 3에서 식별된 액터와 객체들로 구성된

것이어야 하며, 새로운 객체가 추출될 경우 단계 2가 반드시 수행되어야 한다.

국제 기능점수 사용자 그룹(IFPUG)의 측정 수행 지침(Counting Practice Manual)에 정의된 트랜잭션의 각 유형을 살펴보면, 우선 외부 입력(EI)은 어플리케이션 안으로 들어오는 데이터 혹은 제어 정보의 처리를 나타내며, 외부 질의(EQ)는 데이터 혹은 제어 정보의 검색을 통해 어플리케이션 밖으로 결과 데이터를 보내는 것을 나타낸다. 마지막 유형인 외부 출력(EO)은 프로세싱 로직을 통해서 생성된 데이터를 어플리케이션 밖으로 보내는 것을 나타낸다. 외부 출력은 수학식이나 계산과 같은 로직을 최소한 하나 이상을 포함해야 한다. 주목할 점은 외부 입력이나 외부 출력과는 달리 외부 질의는 수학식이나 계산과 같은 로직을 포함하지 않는다는 것이다. 이러한 특성들을 바탕으로 본 논문에서는 다음과 같은 트랜잭션 유형 결정의 규칙을 제안한다.

시퀀스 다이어그램 상에서 시작점을 액터로 하는 메시지 전달 과정이 중단 없이 순차적으로 진행되어서 제어초점(focus of control) 별로 그룹화 되는 것을 하나의 트랜잭션으로 결정한다

- 객체를 생성하는 것과 주어진 인자를 대응하는 객체의 변수에 삽입하는 수정자(modifier)는 외부입력(EI)이다.
- 객체의 변수값을 읽어오는 접근자(accessor)는 외부질의(EQ)이다.
- 객체의 변수값이나 전달된 인자를 가지고 하나 이상의 연산을 수행하거나 비즈니스 로직을 수행하여 새로운 값을 반환하는 것은 외부출력(EO)이다.

복잡도의 계산

식별된 트랜잭션의 유형이 결정되었으면 트랜잭션의 복잡도를 계산하여야 한다. 국제 기능점수 사용자 그룹(IFPUG)의 측정 수행 지침(Counting Practice Manual)은 복잡도 계산을 위해서 데이터 요소 타입(DET)과 참조 파일 타입(FTR)의 수를 식별할 것을 지시한다. 따라서 우리는 본 단계에서 DET와 FTR, 그리고 제시된 세부규칙에 대응하는 UML의 요소들을 식별하여 계산을 수행한다.

데이터 요소 타입은 사용자가 인식 가능하고 유일하며, 반복되지 않는 필드나 속성을 정의하는 것이며 이것은 객체의 속성에 대응한다고 할 수 있다[Uemura, 1999]. 그리고 참조 파일 타입은 트랜잭션에 의해 유지되거나 읽히는 파일을 정의하는 것으로서 식별된 트랜잭션 내에 포함된 객체들에 대응한다. 이러한 대응관계에 기반 한 계산 규칙은 다음과 같다.

- 외부 입력(EI)의 경우 트랜잭션 내에서 액터로부터 전달되는 메소드에 정의된 인자의 총 합을 DET로 한다.
- 외부 질의(EQ)의 경우 트랜잭션 내에서 액터로부터 전달되는 메소드의 인자의 수와 질의결과로 반환되는 데이터 항목 수의 총합을 DET로 한다.
- 외부 출력(EO)의 경우 트랜잭션 내에서 액터로부터 전달되는 메소드의 인자의 수와 반환되는 데이터 항목의 수의 총합을 DET로 한다.

- 트랜잭션의 수행 중 발생하는 다양한 메시지를 범주로 구분하여 하나의 범주를 하나의 DET로 계산한다.
- 각 트랜잭션에 관련되는 모든 객체를 각각 하나의 FTR로 계산한다.

단계 5: 유스케이스 포인트의 계산

단계 4까지의 과정은 국제 기능점수 사용자 그룹(IFPUG)의 측정 수행 지침(Counting Practice Manual)에 정의된 단계를 따라 대응되는 UML의 요소들을 식별하여 적용한 것이다. 따라서 앞서 정의된 단계를 통해서 UML을 적용한 객체지향 시스템의 미조정 기능점수를 추출할 수 있다. 전통적인 기능점수분석에 있어서는 추출된 미조정 기능점수와 값 조정인자를 통해서 전체 시스템의 조정기능점수를 식별한다. 하지만 본 연구에서는 전체 시스템의 기능점수를 측정하는 것과 더불어 시스템의 기능을 표현하는 각 유스케이스의 기능점수를 측정하는 것을 그 목적으로 하고 있다. 하지만 유스케이스의 특성 상 측정된 기능점수에서 직접 유스케이스 당 기능점수를 측정할 수 없다.

유스케이스는 개발 중인 시스템이 수행해야 할 기능을 나타낸다. 이 유스케이스의 기능을 구현하기 위해서는 하나 이상의 객체가 요구된다. UML에서 이러한 관계를 도식화하는 것이 협력(collaboration)이며, 유스케이스와 협력은 실체화(realization) 관계를 가진다. 즉, 유스케이스 모델의 유스케이스들은 협력으로 추적되며 협력은 협력을 수행하기 위한 클래스들의 집합으로 추적된다. 그러나 하나의 클래스는 다수의 협력에 참여할 수 있다. 때문에 유스케이스 당 기능점수를 측정하기 위해서는 각 유스케이스의 실현을 위해서 요구되는 객체와 유스케이스 간 중복되는 객체를 식별하는 단계를 수행해야 한다.

4. 사례연구

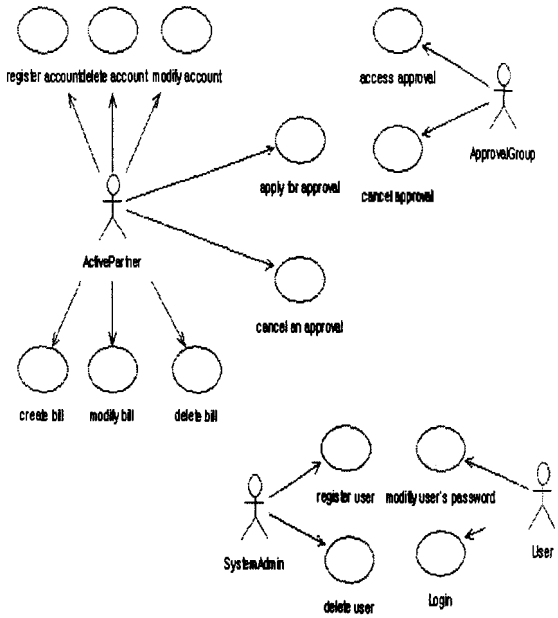
본 장에서는 본 논문에서 제시한 방법을 재무회계 시스템의 일부에 적용해보고, 이를 바탕으로 제안된 방법의 적절성을 검토해본다.

4.1 실험

재무회계 시스템을 구축하는데 있어서, 요구사항 분석단계에 있어서 본 논문에서 제안된 방법을 적용한다. 제안된 방법의 각 단계별 수행 결과는 다음과 같다.

단계1. 본 사례는 시스템을 새로이 구축하는 것을 상정하고 있다. 따라서 기능점수 계산유형은 '개발프로젝트'이다.

단계2. 본 단계의 수행은 산출물 중 유스케이스 다이어그램을 통해서 완료될 수 있다. <그림 2>는 산출된 유스케이스 다이어그램을 나타내고 있다. 본 사례의 경우 모든 액터가 유스케이스에 행동을 요청하고 있다. 따라서 본 사례에 나타난 모든 액터는 전통적 기능점수의 사용자와 일치하며, 계산 범위는 다이어그램 상의 모든 유스케이스를 포함한다.

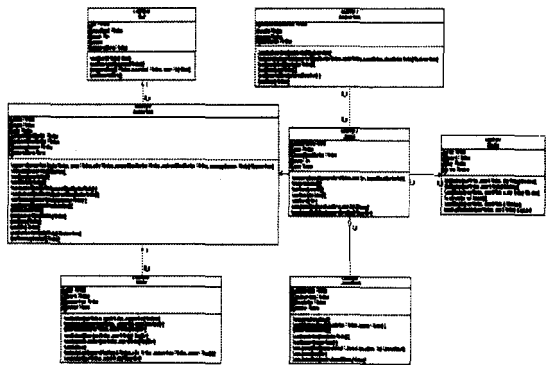


<그림 2> 유스케이스 다이어그램

단계3. 데이터 기능을 계산을 수행해야 하는 본 단계를 위해서는 식별된 클래스가 요구 되며, 이를 위해서 클래스 다이어그램을 사용한다. 본 단계에서 사용된 클래스는 시스템이 관리해야 할 데이터를 나타내는 엔티티 클래 스이다. 다음의 <그림 3>은 본 사례의 클래스 다이어그램을 나타내며, <표 1>은 식별된 클래스를 바탕으로 유형을 결정하고 복잡도를 계산한 결과이다.

단계4. 본 단계에서는 트랜잭션 기능을 계산 하기 위하여 산출물 중 유스케이스 다이어그램과 시퀀스 다이어그램을 사용한다. <그림 4>는 시퀀스 다이어그램을 나타내고, <표 2>는 각 유스케이스가 가지는 트랜잭션의 값을 나타낸다.

단계5. 본 단계는 앞서 계산된 각 계산 값을 통하여 각 유스케이스의 기능점수를 산출하는 단계이다. 단계3에서 계산된 각 클래스의 결과값은 유스케이스에 중복되어서 나타날 수 있다. 따라서 유스케이스 당 사용되는 각 클래스를 식별하고 이를 바탕으로 유스케이스 당 결과값을 계산한다.

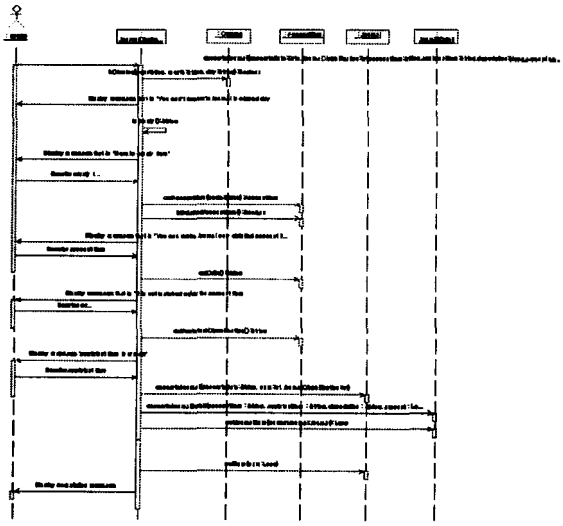


<그림 3> 클래스 다이어그램

<표 1> 클래스의 유형과 복잡도

class \ rule	DET	RET	type
User	5	1	LowILF
AccountItem	8	1	LowILF
Ledger	5	1	LowILF
AssitantItem	4	1	LowILF
Jounal	11	2	LowILF
JounalDetail	5	1	LowILF
Closing	4	1	LowILF

<표 3>은 유스케이스 별로 사용된 클래스를 식별한 후, 이를 바탕으로 유스케이스 별 미조정점수를 계산한 결과이다.



<그림 4> 시퀀스 다이어그램

<표 2> 유스케이스 당 트랜젝션 유형과 복잡도

Use Case	ET	RO	DO	FTR	DET	Type
Login		1		3	1	L
Register user	1	1		6	1	L
Delete user	1	1		4	1	L
Modify user's PW	1	1		4	1	L
Access approval			1	3	4	A
Cancel approval			1	3	4	A
Register account	1		1	8	1	L
Delete account	1		1	3	1	L
Modify account	1	1		7	1	L
Create bill	1	3	1	20	4	H
Modify bill	1	1		8	3	H
Delete bill		1	1	4	3	A
Apply for approval	1	1		3	2	L
Cancel applied approval	1	1		4	2	L

* L : low, A : average, H : high

5. 결론

본 연구를 통해서 우리는 UML을 기저로 한 객체지향개발을 통해 구축된 모델에 기능점수분석을 적용하는 방법을 제안하였다. 본 제안은 구현 언어나 기술에 종속적이지 않은 기능점수의 특징을 바탕으로 하였고, 기능점수분석의 표준인 국제 기능점수 사용자 그룹(IFPUG)의 측정 수행 지침(Counting Practice Manual)과 OMG의 UML1.4를 충실히 따랐다. 따라서 제안된 방법은 기능점수분석을 객체지향 소프트웨어 개발에도 적용할 수 있음을 증명한다고 할 수 있다. 또한 제안된 방법은 유스케이스 모델을 비롯한 객체지향 프로세스의 초기에 구축되는 모델들을 통해서 기능점수분석을 수행하고자 하였다. 이를 통해서 구축하고자 하는 시스템의 규모 측정을 비교적 프로젝트 초기에 수행할 수 있게 한다. 이러한 소프트웨어 생명주기 초기에 규모를 예측하는 것은 인력투입, 비용 산정 등과 같은 중요한 지표를 수립하는데 효과적으로 사용될 수 있으며, 이를 통해서 보다 체계적이고, 합리적인 계획 수립을 가능케 한다. 그러나 제안된 방법의 단점은 우선 제안된 방법을 통해서 산출된 값이 미조정기능점수라는 점이다. 미조정기능점수는 프로젝트의 특성이나 환경을 나타내는 조정인자 값과의 연산을 통해서 조정기능점수를 산출하는데, 본 논문에서는 이러한 조정인자 값에 관한 연구는 수행치 못했다. 따라서 향후 연구과제는 본 논문에서 제안된 방법을 통해서 여러 사례연구를 수행하여 현실적인 조정인자 값을 도출하여 보다 정확하고 객관적인 기능점수 산출을 통한 규모예측을 가능케 하는 것이다.

<표 3> 유스케이스 별 사용클래스

Use Case \ Class	User	Account-Item	Login	Assitna-ttem	Jounal	Jounal-Detail	Closing	CUAP	TUAP	TTUAP
Login								7	3	10
Register user								7	6	13
Delete user								7	6	13
Modify user's PW								7	6	13
Access approval								28	5	33
Cancel approval								28	5	33
Register account								7	7	14
Delete account								7	7	14
Modify account								7	6	13
Create bill								28	33	61
Modify bill								21	13	34
Delete bill								21	9	30
Apply for approval								14	6	20
Cancel applied approval								14	6	20

* CUAP(Class Unadjusted Point), TUAP(Transaction Unadjusted Point), TTAP(ToTal Unadjusted Point)

참고문헌

[서예영, 이남용, 2001] Use Case 다이어그램에 의한 객체지향 소프트웨어 규모 예측 방법에 의한 연구, 정보과학회 2001년 춘계학술대회, 2001

[Booch, 1998] Grady Booch, James Rumbaugh, Ivar Jacobson, The Unified Modeling Language User Guide, 1998

[Chessman, 2000] John Chessman, John Daniels, UML Components, 2000

[Roger S.] Roger S. Pressman. Software Engineering: A Practitioner's Approach 5th. McGraw-Hill.

[Armour, 2001] Frank Armour, Granville Miller, Advanced Use Case Modeling, Addison Wesley, 2001

[Fioravanti et al. 2001] Fioravanti, F.; Nesi, P., Estimation and prediction metrics for adaptive maintenance effort of object-oriented systems, Software Engineering, IEEE Transactions on , Volume: 27 Issue: 12 , Dec. 2001 , Page(s): 1062 -1084

[Kemerer et al, 1992] Kemerer, C.F.; Porter, B.S., Improving the reliability of function point measurement: an empirical study Software Engineering, IEEE Transactions on, Volume: 18 Issue: 11 , Nov. 1992 Page(s): 1011 -1024

[Uehara et al, 1999] Uehara, S.; Mizuno, O.; Kikuno, T., A straightforward approach to effort estimation for updating programs in object-oriented prototyping development, Software Engineering Conference, 1999. (APSEC '99) Proceedings. Sixth Asia Pacific, 1999, Page(s): 144 -151

[Janaki et al, 2000] Janaki Ram, D.; Raju, S.V.G.K., Object oriented design function points , Quality Software, 2000. Proceedings. First Asia-Pacific Conference on, 2000, Page(s): 121 - 126

[Fetcke et al, 197] Fetcke, T.; Abran, A.; Tho-Hau Nguyen , Mapping the OO-Jacobson approach into function point analysis ,Technology of Object-Oriented Languages and Systems, 1997. TOOLS 23. Proceedings, 1998, Page(s): 192 -202

[Uemura et al, 1999] Uemura, T.; Kusumoto, S.; Inoue, K. , Function point measurement tool for UML design specification ,Software Metrics Symposium, 1999. Proceedings. Sixth International, 1999 .Page(s): 62 -69

[Kusumoto, 2002] Kusumoto, S.; Imagawa, M.; Morimoto, S.; Matsusita, K.; Inoue, K.; Tsuda, M. , Function point measurement from Java programs ,Software Engineering, 2002. ICSE 2002. Proceedings of the 24rd International Conference on, 2002, Page(s): 576 -582

[Furey, S., 1997] Furey, S., Why we should use function points [software metrics], IEEE Software, Volume: 14 Issue: 2, Mar/Apr 1997, Page(s): 28, 30

[Abran, A., 1996] Abran, A.; Robillard, P.N., Function points analysis: an empirical study of its measurement processes, Software Engineering, IEEE Transactions on, Volume: 22 Issue: 12, Dec. 1996, Page(s): 895 -910

[Norman et al, 1997]Norman E. Fenton, Shari Lawrence Pfleeger. Software Metrics: A Rigorous and Practical Approach 2nd. INTERNATIONAL THOMSON COMPUTER PRESS.

[Kitchenham, 1997]Kitchenham, B, The Problem with Function Points, IEEE Software, Volume: 14 Issue: 2, Mar/Apr 1997, Page(s): 29 -31

[Low et al, 1990] Low, G.C.; Jeffery, D.R. , Function points in the estimation and evaluation of the software process ,Software Engineering, IEEE Transactions on , Volume: 16 Issue: 1 , Jan. 1990 ,Page(s): 64 -71

[Basili et al, 1996] Basili, V.R.; Briand, L.C.; Melo, W.L. , A validation of object-oriented design metrics as quality indicators ,Software Engineering, IEEE Transactions on , Volume: 22 Issue: 10 , Oct. 1996 ,Page(s): 751 -761

[Kirsten, 2001] Kirsten Ribu, Estimating Object-Oriented Software Projects with Use Cases, University of Oslo Department of Informatics, 2001

[Gustav Karner, 1993] Gustav Karner, Use Case Points - Resource Estimation for Objectory Projects, Objective Systems SF AB (copyright owned by Rational Software), 1993

[John] John Smith, The Estimation of Effort Based on Use Cases, Rational Software white paper,

[IFPUG, 2000] International Function Point Users Group(IFPUG), Function Point Counting Practices Manual, Release 4.1., IFPUG, Westerville, Ohio, 2000

저자소개

안계중 (e-mail : ahnbe@beans.soongsil.ac.kr)
2001년 숭실대학교 소프트웨어공학과 졸업(공학사)하고, 2002년~현재 숭실대학교 일반대학원 컴퓨터학과 소프트웨어공학 전공을 하고 있다. 관심분야는 객체지향 개발방법론, 컴포넌트 기반 방법론, Business-Modeling 등이다.

이남용
숭실대학교 컴퓨터학부에서 공학사를, 고려대학교에서 경영학(MIS) 석사를, 미시시피주립대학교 경영학 박사학위(MIS)를 취득하였다. 국군정보사에서 정보시스템분석 장교로 근무(ROTC#17)하고 대위로 전역 후 한국국방연구원 (KIDA), 국방정보체계연구소(KIDIS), 국방과학연구소(ADD)등에서 20 년 간 연구책임자, 연구부장 등으로 근무하였다. 현재 숭실대학교 정보과학대학 컴퓨터학부 (소프트웨어공학) 교수로 재직하고 있으며, 현재, 한국전자거래학회 논문편집위원장, 정보과학회, 정보처리학회, 정보보호학회, 산업정보학회등에서 학술위원으로, 국회정보통신포럼, 국방부, 산자부, 정통부, 중소기업청, 경찰청, 서울시청 등에서 자문교수로 봉사하고 있다. 또한, 저자는 바산네트워크(주) 대표이사도 겸직하고 있다. 2001 년 설립된 바산네트워크(주)는 객체기술(UML, RUP), 컴포넌트기술(OOAD, CBD), 소프트웨어 테스팅기술(Functionality, Performance, Reliability Test)을 리셀러, 맞춤교육, 컨설팅하는 전문기업이며, 2002.1.21 일에 기술신용보증기금(재)으로부터 우수적격 기술평가를 받아 중소기업청에 벤처기업으로 등록하였다.
저서로는 Ada 언어와 응용(정익사, 1987), CALS/EC(법영사, 1996), 인스턴트코바 (영진출판사,1998), 칼스와 전자상거래(법영사, 1999), 소프트웨어 프로세스(라메르정보기술, 2000), 전자상거래시스템론(법영사, 2001)외 다수가 있으며, 논문으로는 " An Empirical Study On Software Reuse" , *IEEE Transactions on Software Engineering*, Vol.23, No.9, September 1997 외 80 여 편을 발표하였다.