「정보시스템연구」 제16권 제1호
한국정보시스템학회
2007년 3월, pp. 111~133

객체지향 기반의 정보시스템 개발 프로젝트에서의 기능점수 예측 기법에 관한 연구

정승렬* · 이석준**

〈목 차〉

Ⅰ. 서 론

현대 경영환경에서 정보기술의 활용은 중요한 전략적 차원으로 활용되고 있으며 조직의 환경과 특성에 맞는 정보시스템의 확보는 조직의 성과를 높이는데 매우 중요한 역할을 한다. 조직에서 하나의 정보시스템을 개발한다고 하는 것은 사용자 요구사항의 파악으로부터 시작하여 구현된 시스템이 실제 업무에 활용되기까지 길고도 복잡한 과정을 거친다. 그리고 이 과정은 많은 사람들의 참여와 다양한 도구 및 기술의 활용을 필요로 한다.

정확한 정보시스템 개발비용의 예측은 향후 개발될 정보시스템에 대한 정확한 자원배분을 가능하게 하며 이는 시스템의 품질에 결정적인 영향을 준다. 따라서 정보시스템 개발에 소요되는 비용과 자원, 기간 등의 예측은 해당 프로젝트를 성공적으로 수행하기 위한 필수적 요소라고 할 수 있으며, 이러한 예측 활동은 프로젝트의 초반에 수행되어야 보다 효과적일 수 있다.

* 국민대학교 비즈니스IT학부 교수, srjeong@kookmin.ac.kr
** SK C&C 금융사업부 과장, junelee@skcc.com

이러한 이유로 정보시스템 개발을 위한 프로젝트의 초기 단계에서 정확한 정보시스템의 개발 비용을 예측하는 활동이 점차로 중요시 되고 있다(Pressman, 1997).

정보시스템 개발비용 측정의 주요 인자로는 정보시스템의 개발규모와 환경 등을 들 수 있는데 이 두 가지의 인자 중에서 정보시스템 개발 비용에 영향을 미치는 가장 핵심적인 요소는 개발규모라고 할 수 있다. 정보시스템 개발규모를 측정하는 기법으로 가장 널리 사용되는 방법은 프로그램 라인 수(LOC, Line of Code) 측정 방식과 기능점수 분석기법(FPA, Function Point Analysis)이 있다(Putnam, 1992; Dolado, 1997).

우리나라에서는 정보시스템 개발 및 유지보수 사업의 규모 산정에 주로 프로그램 라인 수 측정 방식을 사용해 왔는데, 이는 프로그램 소스 코드의 라인 수를 통해 소프트웨어의 규모를 산정하는 방법으로 개발자가 이해하기 쉽고 의미가 명확하기 때문에 지금까지 가장 널리 사용되어 왔다. 하지만 프로그램 라인 수 측정 방식은 정보시스템 개발 프로젝트의 초기에 라인 수를 예측하기 어렵고 정보시스템 개발 언어나 환경에 영향을 받으므로 규모예측 방법의 근거가 희박하며 추산 시점에 하자가 발생할 수밖에 없는 등의 여러 가지 문제점을 가지고 있다(Kemerer and Porter, 1992; Matson et al., 1994).

반면에 기능점수(FPA) 분석기법은 사용자에게 제공하는 기능이 많은 시스템의 규모가 기능이 작은 시스템보다 클 것이라는 가정에서 출발하여 시스템이 사용자에게 제공하는 기능을 추정함으로써 소프트웨어의 규모를 예측하는 방법이다. 기능점수 모형은 사용자 관점에서 정보시스템이 제공하는 기능을 중요한 요인으로 분석하고 이를 정량화하여 규모를 측정하기 때문에 사용자가 이해하기 쉽고 프로그램 개발 언어와 환경의 특성과는 무관하게 측정할 수 있다는 장점이 있다(Furey, 1997; Orr and Reeves, 2000).

정보시스템 비용측정 방식의 최근 동향은 프로그램 라인 수 측정 방식과 같은 개발자 관점의 접근방식 보다는 사용자의 관점에서의 경제적 가치를 고려하는 기능점수 분석기법이 보다 선호되고 있다(Abran and Robillard, 1996). 이와 더불어 최근에는 기능점수를 정보시스템 개발 초기에 적용하여 기능점수를 예측하고자 하는 활발한 활동이 일어나고 있으며 대표적인 기능점수 예측 방법론으로는 NESMA의 IFPC (Indicate Function Point Count) 및 EFPC (Estimated Function Point Count) (NESMA, 2004), ISBSG의 EPFS (Early Prediction of Function Size) (ISBSG, 2004), COSMIC의 COSMIC-FFP (Symons, 2003) 등이 있다.

하지만 이러한 연구들은 정보시스템 개발 초기에 비교적 적은 노력으로 기능점수를 예측할 수 있다는 장점이 있으나 지나치게 단순화시킨 기능점수 예측 방식으로 인해 최종 산출물에 대한 기능점수와 예측기법의 기능점수간의 측정 오차 또한 상당수 존재하며, 기능점수 분석의 기초 자료가 되는 기능점수 측정규칙 및 측정요소 자체가 최근의 정보시스템 개발 환경인 객체지향 통합 모델링 언어(UML, Unified Modeling Language)나 객체지향 통합 프로세스(UP, Unified Process)에 직접적으로 적용시키기에는 여러 가지 어려움이 존재한다(Kusumoto *et al.*, 2000; Ribu, 2001; Uemura *et al.*, 2001).

본 연구는 이러한 어려움을 극복하기 위해 먼저, 객체지향 기반의 정보시스템 개발 프로젝트에서 보다 정확하게 기능점수를 예측할 수 있는 기능점수 예측기법을 제시하고자 한다. 이를 통하여 정보시스템에 대한 정확한 자원배분이 가능하게 되고 결과적으로 정보시스템 개발 프로젝트를 성공적으로 수행하는데 기여하게 된다.

제시된 객체지향 시스템의 기능점수 예측기법에 대해서는 UML 및 UP의 산출물을 이용하여 그 적정성을 실증적으로 검증하고자 한다. 이를 위해서는 정보시스템 개발 프로젝트의 초기에 적용이 가능한 기존의 다른 기능점수 예측기법들과 그 정확성을 비교 분석하고자 한다.

Ⅱ. 이론적 배경

2.1 정보시스템 규모 측정

정보시스템 규모의 측정은 시스템 개발 프로세스를 개선하게 하며 계획 수립을 돕고 프로젝트를 통제할 수 있게 함으로써 생산되는 정보시스템의 기능 및 품질을 향상시킨다(Park et al., 1996). 정보시스템의 규모를 측정하는 방법으로 가장 널리 사용되는 방법으로는 프로그램 라인수(LOC) 측정 방식과 기능점수 분석기법(FPA)이 있다. 과거에는 정보시스템 개발 및 유지보수에 대한 규모 산정을 위해 주로 LOC 측정 방식을 사용해 왔는데, 이는 개발자 중심적인 접근 방식으로 평가자의 축적된 경험과 데이터에 따른 추측에 의존하므로 개인차가 발생하기도 하고 이를 구현하는데 있어서 재현성이 낮다는 문제가 있다. 뿐만 아니라 LOC 측정 방식은 동일한 기능을 수행하는 프로그램을 개발하는 경우에도 개발언어에 따라서 프로그램 LOC의 편차가 크게 나타나는 개발환경의 종속성 문제라든지(Matson et al., 1994), LOC 자체가 정보시스템 개발 초기에 적용하기가 매우 어렵고 개발이 완료된 후에야 측정이 가능한 측정시점의 문제라든지(Emrick, 1987), LOC 측정 방식에 대한 측정 기준이 모호하여 변형된 라인 수 계산 방법에 따라 매우 큰 편차가 발생되는 측정의 불확실성 문제가 존재하기도 한다(Jones, 1988). 그 외, LOC 측정방식은 단지 길이만을 고려하고 있어, 정보시스템의 기능성 또는 복잡도를 반영하지 못하는 문제점 등을 가지고 있다(Kemerer and Porter, 1992).

이러한 프로그램 LOC 측정 방식의 단점을 보완하고자 정보시스템의 기능성과 복잡도에 대한 광범위한 연구가 이루어졌으며, 정보시스템의 기능성으로 정보시스템 개발규모나 개발노력을 산정하는 FPA가 제안되었다. Albrecht (1979)가 최초로 제안한 FPA는 사용자의 관점에서 정보시스템이 제공하는 기능의 수를 측정하여 정보시스템의 규모를 측정하는 방법으로서 논리적 기능단위를 이용하여 정보시스템의 개발 규모를 기능점수로 측정한다. 이러한 FPA는 정보시스템의 개발 언어나 환경에 영향을 받지 않으며 개발전문가 뿐만 아니라 사용자까지도 충분히 공감할 수 있고 산정결과를 정확한 개발노력으로 환산될 수 있다는 장점이 있다(Furey, 1997; Orr and Reeves, 2000)

Albrecht가 기능점수 모형을 제안한 이후, 이를 보완하고 조직에 맞게 수정한 여러 가지 변형된 기능점수 모형이 제안되었다. 기능점수모

형은 국제 기능점수 사용자 그룹(IFPUG, International Function Point User Group)이 제시한 국제 표준이지만 기능점수 예측을 위해 이를 응용한 NESMA(2004), ISBSG(2004)등의 기능점수 측정기법들이 제안되고 있고 실시간 소프트웨어와 다계층 소프트웨어의 기능을 측정할 수 있는 COSMIC-FFP (Symons, 2003) 모형도 ISO 인증을 받으면서 국제 표준으로 승인되는 등 최근까지도 기능점수 측정기법과 관련한 연구가 활발하게 수행되고 있다.

2.2 유스케이스(Use Case) 모델

대부분의 정보시스템 개발 프로젝트의 실패 원인은 개발 초기에 사용자 요구사항 도출의 어려움에 기인한다. 이는 사용자 요구사항이 정형화된 규칙 없이 여러 문헌에서 보여주고 있는 애매모호한 서술적인 개념과 몇몇 단편적인 조언을 바탕으로 각 요소를 추출하기 때문이다(유철중, 정소영, 2002). 이러한 문제점을 해결하기 위한 방안으로 Jacobson et al.(1992)에 의해 요구사항을 도출하기 위한 방법으로 유스케이스(Use Case)가 최초로 제안되었으며, UML(Ambler, 1998)에 유스케이스 다이어그램(Use Case Diagram)이 포함되면서부터 객체기반의 정보시스템 개발 프로젝트에는 UML 및 UP의 수행과정으로 유스케이스 모델이 널리 활용되고 있다.

UML 및 UP는 유스케이스 중심의 개념을 가지고 있어 개발자는 시스템의 기능적 요구사항들을 결정하고 시스템이 무엇을 할 것인가에 대한 상세한 설계로서 유스케이스 모델링을 수행하게 된다(Schach, 2004). 이 과정에서 유스케이스는 시스템의 행위를 결정하는 것으로 시스템의 기능을 정의하고 범위를 결정함으로써 시스템과 외부 환경 변수를 구분하고 상호 관계를 정립하는 기능을 가지고 있다. 또한 유스케이스 모델은 유스케이스와 액터(Actor)와의 관계를 UML로 시각화하여 유스케이스 다이어그램을 작성함으로써 사용자 및 고객에게 구체적인 정보를 빠르게 전달할 수 있다(Ambler. 1998).

유스케이스는 그 형식에 따라서 개요수준, 분석수준, 설계수준으로 구분할 수 있으며 객체지향 시스템 개발에 있어 가장 중추적인 역할을 한다(Larman, 2002). 유스케이스 모델은 시스템 개발의 근본으로 클래스를 인식하는 원천이면서 시스템 분석 및 설계, 그리고 시험 단계까지를 전체적으로 제어한다. 기능적 방법론에서는 요구사항이 정의된 후, 구현단계에 가면 그 추적이 어려워지는 경향이 있지만, 유스 케이스는 프로젝트가 진행되어 감에 따라 발주자에게 인수에 필요한 추적성을 제공하며 개발자에게는 정보시스템 개발의 시작 문서로서의 활용성을 인정받고 있다. 결국, 유스케이스는 사용자의 요구가 다음 단계의 모형에 어떻게 반영되고 있는지를 추적할 수 있도록 하는 통제도구 역할을 한다(정승렬, 임좌상, 2004).

Ⅲ. 유스케이스 기능 점수

3.1 도출 과정

본 연구에서는 객체지향 기반의 기능점수 측정 방안이 유스케이스를 기반으로 한다는 점에서 이를 유스케이스 기능점수(UCFP, Use Case

Function Point) 분석기법이라 명한다.

IFPUG의 기능점수 측정기법에서는 측정대상 어플리케이션 경계내의 데이터 그룹 또는 제어정보를 데이터 기능(Data Function)으로 측정하며, 어플리케이션 경계를 넘나드는 데이터나 제어정보를 처리하는 단위 프로세스를 트랜잭션 기능(Transaction Function)으로 측정한다. 기능점수 측정기법에서는 기능점수를 계산하는데 존재하는 주관적인 판단이 평가자별로 크기 때문에 기능점수의 편차가 발생할 수 있다. 이러한 문제를 해결하기 위해 IFPUG에서는 기능점수 측정 매뉴얼(FPCPM: Function Point Count Practice Manual)을 만들어 지속적으로 개정하고 있다.

한편, Kusumoto *et al.*(2000)과 Uemura *et al.*(2001)은 IFPUG의 기능점수 측정기법의 분석대상을 수정하여 제안하였는데, 기능점수 측정단위를 어플리케이션이 아닌 클래스(Class)로 변경하여 객체지향의 특성을 기능점수 측정에 반영시켰다. 또한 데이터 기능을 측정하기 위한 대상으로 객체지향 분석기법의 산출물인 순차 다이어그램(Sequence Diagram)을 사용하며, 트랜잭션 기능을 측정하기 위한 대상으로는 순차 다이어그램(Sequence Diagram)에 존재하는 메시지(Message)를 사용함으로써, UML의 동적인 관점에서 기능점수를 측정하는 방법이다.

Kusumoto et al.의 기능점수 측정기법은 이후에 Shinji, *et al.*(2002)에 의해 확장 연구되었는데, 제시된 주요 내용은 트랜잭션 기능을 측정하기 위한 대상으로 사용되었던 메시지(Message)를 대신하여, 자동화 도구를 사용하여 프로그램의 소스 코드를 분석함으로서 기능점수를 측정할 수 있는 방안을 제시하는 것이었다. 하지만 Shinji, *et al.*의 연구는 프로그램의 소스 코드가 모두 생성되는 시점, 다시 말해 정보시스템의 구현이 완료된 시점 이후에나 적용이 가능하다는 문제점이 있다.

이러한 문제점을 해결하기 위한 방안으로 본 연구에서 제안한 기능점수 측정기법의 분석대상은 다음과 같다. 유스케이스 기능점수는 IFPUG의 기능점수 측정기법에 객체지향의 특성을 반영하기 위하여 Kusumoto et al.이 제안한 기능점수 측정기법과 같이 기능점수 측정 단위를 클래스 단위로 선정하며 데이터 기능과 트랜잭션 기능의 분석을 위하여 UML의 주요 산출물인 클래스 다이어그램과 유스케이스를 사용한다.

본 연구에서 제안한 유스케이스 기능점수 측정기법은 UML의 클래스 다이어그램을 사용하여 시스템의 정적인 정보를 분석하고, 정보시스템의 업무 프로세스 정보가 담겨져 있는 유스케이스의 분석을 통해 시스템의 동적인 정보를 측정한다. 유스케이스 기능점수 측정기법의 주요 분석단위와 대상인 클래스, 클래스 다이어그램, 유스케이스는 UP의 초기단계인 요구정의 단계 이후부터 도출되기 시작하며, 이는 정보시스템 개발 프로젝트 초기부터 기능점수의 예측이 가능하다는 것을 의미한다.

3.2 측정 범위

IFPUG의 기능점수 측정 실무 지침(FPCPM, Release 4.2)에 따르면 기능점수 측정 절차는 총 6단계를 거쳐서 수행된다.

기능점수의 측정을 위해서는 우선 기능점수

측정 유형을 결정하고, 기능점수 측정의 범위(Scope)와 어플리케이션의 경계(Boundary)를 식별한 후 기능점수를 데이터 기능과 트랜잭션 기능의 2가지 기준으로 분류하여 측정한다. 측정된 기능점수들을 모두 합한 값이 미조정 기능점수(UFP, Unadjusted Function Point)이며 여기에 일반시스템의 특성(GSC, General System Characteristics)을 반영하여 조정인자(VAF, Value Adjustment Factor)를 산출하고 이를 통해 최종적으로 조정 기능점수(AFP, Adjusted Function Point)를 산출한다.

IFPUG의 기능점수 측정 절차 6단계 중 제 5, 6단계에 해당하는 조정인자(VAF)의 결정과 조정 기능점수(AFP)를 산출하는 과정을 살펴보면, 조정인자의 결정은 정성적 평가 기준에 따라 측정자의 주관적 판단에 의존하도록 되어 있으며 이로 인하여 동일한 조직의 사람들도 동일한 프로젝트에 대한 조정인자 등급을 다르게 판단하는 경향이 있다(Low and Jeffery, 1990). 또한 조정인자를 통해 산출되는 조정 기능점수의 설명력 향상이 크지 않은 것으로 나타나고 있으며(Kemerer and Porter, 1992; Lokan, 2000), 이러한 문제점으로 인해 조정 기능점수는 실무에서 널리 활용되지 못하고 있다(Ribu, 2001).

선행 연구들을 통해 밝혀진 이러한 문제점 이외에도 조정인자의 산출 근거인 일반시스템 특성은 구현된 정보시스템이 가지는 특성이므로 기능점수 예측의 목적을 가지고 있는 본 연구의 목적과는 일치하지 않으며 이는 정보시스템 예산 집행의 표준인 정보통신부의 소프트웨어 사업대가 산정기준에서도 미조정 기능점수를 기준으로 하는 측면과 일치한다(정보통신부, 2005).

따라서 본 연구에서는 IFPUG에서 제안한 FPA의 측정 절차 중 조정인자의 도출 및 이를 통한 조정 기능점수 측정 단계를 분석대상에서 제외하였으며, 유스케이스 기능점수의 분석 범위는 정보시스템 개발 초기단계에 적용될 수 있는 미조정 기능점수의 측정까지를 그 분석 범위로 결정한다.

3.3 예측시점 및 측정 모형

본 연구에서는 정보시스템 개발 프로젝트의 진행에 따른 기능점수 예측시점을 두 군데 즉, UP의 착수 단계(Inception Phase) 한 군데와 상세 단계(Elaboration Phase)의 말기 한 군데로 구분하고, 각각의 예측시점에 적용 가능한 2가지의 유스케이스 기능점수 예측기법 즉, $UCFP_1$과 $UCFP_2$를 제안한다. 이들에 대한 상세한 내용은 다음과 같다.

3.3.1 유스케이스 기능점수-I (UCFP1)

유스케이스 기능점수-I($UCFP_1$)은 정보시스템 개발의 착수 단계에서 적용될 수 기법이다. 이는 정보시스템 개발 초기에 사용자 요구사항이 도출되는 과정에서 생성된 초기의 유스케이스를 통해 기능점수를 분석하는 방법으로, 기능의 개수가 도출되고 레코드요소 유형(RET)과 참조파일 유형(FTR)은 도출 될 수 있으나 데이터요소 유형(DET)이 도출되지 않은 상태에서 적용될 수 있다. $UCFP_1$은 기능점수 측정을 위하여 트랜잭션 기능을 액터와 참조파일 유형(FTR)의 수에 따라 기능점수를 예측하며, 데이터 기능은 NESMA의 EFPC와 같이 평균 복잡도를 통해 기능점수를 예측하는 방법이다.

$UCFP_1$에 대한 기능점수 측정 규칙은 유스케이스 기능점수 예측기법에 대한 측정 규칙을 도출한 후, 이를 실제 사례에 적용함으로서 얻어지는 액터와 참조파일 유형간의 패턴을 분석함으로써 도출될 수 있다.

3.3.2 유스케이스 기능점수-II (UCFP2)

유스케이스 기능점수-II($UCFP_2$)는 정보시스템 개발의 상세 단계에서, 시스템 설계 작업의 비중이 높아지는 시점에서부터 구축 단계 이전까지의 시점에 적용될 수 있다. 이는 UP의 산출물인 유스케이스 기술서(Use Case Description)와 클래스 다이어그램만을 사용하여 기능점수를 예측하는 방법으로서 정보시스템의 구현이 완료되지 않은 시점에서 기능점수를 예측하는 방법이다.

3.4 유스케이스 기능점수의 측정 규칙

3.4.1 연구모형의 적합도

IFPUG의 기능점수 분석기법에 따른 기능점수 측정절차의 첫 단계는 기능점수 측정유형을 결정하는 것이다. 기능점수의 측정유형은 크게 개발 프로젝트 기능점수(DFP, Development Function Point), 개선 프로젝트 기능점수(EFP, Enhancement Function Point), 어플리케이션기능점수(AFP, Application Function Point)의 3가지 유형으로 분류된다(IFPUG, 2004; Garmus and Harron, 2001).

DFP와 EFP는 프로젝트의 수행 목적에 따라 구분되는 것으로서, DFP는 프로젝트를 통해 새로 개발된 정보시스템 전체를 기능점수 측정대상으로 하며 EFP는 프로젝트를 통해 변경된 기능만을 기능점수 측정대상으로 한다는 차이점이 있다. 2가지의 프로젝트 기능점수 유형 모두 프로젝트에 대한 사용자 요구사항이 반영된 변환 기능(CFP, Conversion Function Point)을 포함한다. 여기에서 변환 기능이란 정보시스템의 설치 시에 데이터를 변환하기 위한 기능이나 사용자가 규정한 다른 변환 요구들을 제공하기 위한 기능을 의미한다. AFP는 어플리케이션의 설치가 완료된 시점에서 사용자에게 제공되는 기능만을 측정하는 것으로서 변환 기능점수는 포함되지 않는다. 개발 프로젝트의 어플리케이션 기능점수와 개선 프로젝트의 어플리케이션 기능점수의 2가지 유형으로 측정될 수 있으며 개발 또는 개선된 정보시스템 전체를 기능점수 측정대상으로 한다.

본 연구에서 제시하는 유스케이스 기능점수(UCFP)는 원칙적으로 IFPUG 기능점수의 모든 측정 유형에 걸쳐서 적용될 수 있으며, 이는 측정유형의 구분이 기능점수 측정 방법과 관련된 사항이 아니라 기능점수 측정의 범위와 관련된 사항이라는 점에 기인한다. 단, UCFP를 통한 DFP의 미조정 기능점수(UFP) 측정은 프로젝트 초기에 정해진 규칙에 따라 정량적인 기준으로 측정할 수 있는 반면에 개발프로젝트의 조정 기능점수(AFP)를 비롯한 개선 프로젝트 기능점수와 어플리케이션 기능점수는 본 연구의 수행 목적이 정보시스템의 초기에 기능점수를 예측하는데 있음을 고려할 때 적합하지 않다. 특히, 조정 기능점수의 측정을 위해서는 반드시 조정인자 측정의 기준인 일반시스템 특성이 명확히 측정되어야 하며, 이러한 이유로 조정 기능점수 측정은 정보시스템의 상세설계 단계 이후로 한정하는 것이 적절하다. 물론 이때 필요

한 조정인자의 측정은 IFPUG의 기능점수 분석 기법 기준을 따른다. 본 연구에서는 유스케이스 기능점수의 측정유형을 개발 프로젝트의 미조정 기능점수까지로 한정하였으며 이를 정리하면 다음과 같다.

<표 3-1> UCFP의 측정 유형 결정 규칙

	규칙
1	UCFP의 목적은 정확한 미조정 기능점수(UFP)를 예측하는데 있으므로 기능점수의 측정유형을 개발 프로젝트의 미조정 기능점수로 한정한다.
2	UCFP의 개발 프로젝트의 미조정 기능점수의 산출 공식은 다음과 같이 정의한다. UCFP = UFP(미조정 기능점수) + CFP (자료 전환에 의해 포함되는 기능의 기능점수)

3.4.2 측정범위와 어플리케이션 경계식별

기능점수의 측정 범위는 실제 기능점수 계산에 포함되는 기능성을 규정하는 것이며 정보시스템 개발 프로젝트의 활동에 의해 영향을 받는 모든 기능들을 포함한다. 어플리케이션 경계는 측정 대상 정보시스템과 사용자 간의 경계를 의미하는데 IFPUG의 어플리케이션 경계의 식별은 다음과 같은 3가지 규칙을 따른다. 첫째, 경계는 사용자 관점에 기초하여 결정되는 것으로, 그 초점은 사용자가 무엇을 이해하고 기술하느냐에 있다. 둘째, 관련 어플리케이션 간의 경계는 기술적 고려보다는 별도의 비즈니스 영역에 기초한다. 셋째, 어플리케이션에 대해 이미 설정된 최초의 경계는 측정범위에 의해 영향을 받지 않는다.

위에서 언급한 세 가지의 어플리케이션 경계 식별의 규칙은 유스케이스 분석을 통하여 사용자 관점에서 시스템을 분석하는데 이용될 수 있다. UML에 따른 유스케이스 분석의 산출물은 크게 유스케이스 다이어그램과 유스케이스 기술서로 구성되는데 유스케이스 다이어그램은 시스템이 수행하여야 할 행위를 명세화하며 시스템과 서브 시스템, 클래스의 각 요소 행위를 모형화 하는 도구로서 유스케이스와 액터간의 관계를 표현한다. 유스케이스 명세서는 유스케이스의 사건(Event)의 흐름을 파악하고 발생 가능한 시나리오를 추측할 수 있도록 문서화 한 것으로 프로젝트가 진행되면서 개발 문서로서의 활용성을 인정받고 있다(Colbert, 1999; 정승렬, 임좌상, 2004).

IFPUG의 기능점수 분석에 따른 어플리케이션 경계 식별의 첫 번째 규칙에 대하여 유스케이스는 사용자가 시스템을 사용하는 사례를 묶어 놓은 것이기 때문에, 사용자의 관점에서 시스템을 구현할 수 있다는 장점이 있다(Colbert, 1999). 또한 사용자가 추구하는 목표 또는 요구를 충족하기 위해 시스템을 사용하는 사례로서 사용자의 요구사항을 이해하고 이를 기술하는데 탁월한 기법으로 알려져 있다(Larman, 2002).

경계 식별의 두 번째 규칙인 기능점수의 경계가 사용자의 관점에 기초하여 결정되며, 기술적인 고려가 아닌 비즈니스 영역에 의해 결정된다는 것은 유스케이스의 개념적 특징과 많은 유사성을 갖는다. 유스케이스는 사용자 관점에서 시스템을 설계하기 위한 관점으로 작성되며, 이에 따라 기능의 절차를 포함하는 것이 일반적이다. 프로세스의 시작과 끝이 있다는 점에서 프로세스 모델링에서 말하는 기본 업무 프로세스

와 비교되기도 하는데, 프로세스는 사용자의 가치 창출을 시스템 설계의 기준으로 삼고 있다는 점에서 유스케이스와 같은 목적을 가지고 있다고 할 수 있다(Nurcan et al., 1998). 또한 유스케이스는 사용자와 개발자간의 의사소통에 도움이 되며 사용자의 요구사항에 대한 정보를 토대로 비즈니스 프로세스를 표현하는 표준기법으로 시스템의 경계 식별에 도움이 될 수 있다(Ambler, 1998).

경계 식별의 세 번째 규칙은 기능점수 측정범위가 무엇을 측정하는가의 문제로 측정 목적과 대상에 따라 변화할 수 있지만 이미 설정된 어플리케이션 경계는 내부 어플리케이션과 외부 사용자간의 개념적 인터페이스로 측정 대상이나 목적에 영향을 받지 않는다는 것이다. 조사 대상 소프트웨어와 다른 소프트웨어 간의 어플리케이션 경계를 설정하는 일은 주관적일 수 있으며 하나의 어플리케이션이 끝나고 다른 어플리케이션이 시작되는 시기를 명확히 구분하는 일은 매우 어렵다. 따라서 경계의 설정은 기술적 또는 물리적 고려사항에 기초하기 보다는 비즈니스 관점에서 고려하도록 노력해야 하며 어플리케이션의 경계를 설정할 때에는 신중하게 결정하여야 한다. 왜냐하면 경계를 넘나드는 모든 데이터가 잠재적으로 측정 범위에 포함될 수 있기 때문이다(Garmus and Harron, 2003).

본 연구에서는 어플리케이션의 경계를 식별하는 요소로 UML과 UP에서 사용되는 경계 클래스(Boundary Class)와 제어 클래스(Control Class)를 사용하였다. 경계 클래스는 개발할 시스템과 시스템 외부와의 연결, 즉 인터페이스 역할을 하는 클래스들을 의미하며 사용자 인터페이스를 위하여 사용되는 각종 화면이나 외부 시스템과의 연동을 위한 통신기능을 담당한다. 또한 제어 클래스는 시스템이 제공할 유스케이스의 제어 로직 및 비즈니스 로직을 제공하는 것으로 다른 제어 클래스 또는 경계 클래스와의 연동을 위한 역할을 담당한다. 따라서 본 연구에서는 기능점수 측정기법의 어플리케이션 경계 식별을 위하여 유스케이스의 경계 클래스와 제어 클래스를 참조하여 시스템의 경계를 식별하였다. 본 연구에서 사용된 기능점수의 측정범위와 어플리케이션 경계의 식별 규칙을 정리하면 다음과 같다.

<표 3-2> 측정범위와 어플리케이션 경계의 식별 규칙

	규칙
1	기능점수 측정 범위는 측정 목적과 대상에 따라서 변화할 수 있으며, 어플리케이션 경계는 변화하지 않는다.
2	어플리케이션 경계는 물리적, 또는 기술적 고려사항이 아닌 유스케이스에 명시되어 있는 사용자의 비즈니스 관점을 근거로 결정한다.
3	어플리케이션 경계를 식별하기 위하여 클래스 다이어그램의 경계 클래스와 제어 클래스를 참조한다.

3.4.3 데이터 기능의 측정 규칙

1) 데이터 기능의 식별규칙

IFPUG에서는 데이터 기능을 사용자가 식별할 수 있는 논리적으로 연관된 데이터 그룹 또는 제어정보라고 규정하고 있으며 그 유형을 내부 논리 파일(ILF, Internal Logical File)과 외부 연계 파일(EIF, External Interface File)의 두 가지로 분류하고 있다. 내부 논리 파일은 어플

리케이션 경계 내부에서 유지되는 내부 논리 파일을 의미하며, 외부 연계 파일은 다른 어플리케이션의 내부에서 유지되고 측정 대상 어플리케이션이 참조하는 파일을 말한다(IFPUG, 2004).

여기에서 사용되는 파일이라는 개념은 전통적인 데이터 처리 관점의 파일이나 물리적으로 구현된 데이터 그룹이 아닌 논리적으로 관련된 데이터 그룹을 의미하는 것으로서 이러한 파일의 개념은 유스케이스에서 사용되는 요소 중에서 객체 또는 클래스에 해당한다고 할 수 있다(Colbert, 2001). 하지만 데이터와 데이터를 조작하는 처리 작업을 동시에 가지는 클래스의 특징과 추상화가 높아짐에 따라 다양하게 나타나는 클래스의 유형을 고려할 때 클래스의 개념과 기능점수의 파일 개념을 동일한 개념으로 보기에는 무리가 있다. 따라서 본 연구에서는 기능점수에서의 파일의 정의를 클래스의 여러 유형 중 기능점수의 파일의 개념에 상응하는 엔티티 클래스(Entity Class)로 한정한다. 엔티티 클래스는 영속적인 정보와 그 정보에 대한 조작 기능을 제공하는 클래스이며 업무와 작업의 참여자 및 결과물 유형에 해당된다. 이와 같은 클래스의 특성을 고려하여 유스케이스 기능점수의 데이터 기능 측정규칙은 <표 3-3>과 같이 정의할 수 있다.

2) 데이터 기능의 복잡도와 기여도 식별규칙

IFPUG에 따르면 데이터 기능의 유형을 결정한 후, 데이터 기능의 복잡도와 기여도를 통해 데이터 기능에 대한 미조정 기능점수를 측정할 수 있다. 데이터 기능에 대한 복잡도와 기여도는 내부 논리 파일과 외부 연계 파일에 대해 각각의 데이터 요소 유형(DET, Data Element Type)과 레코드 요소 유형(RET, Record Element Type)의 수를 구하여 이에 따라 기능 복잡도를 결정한다.

데이터 요소 유형은 사용자가 식별 가능하고 비 반복적인 유일한 필드를 의미하며 이것은 클래스 다이어그램이 가지는 속성에 대응하는 개념이라고 할 수 있다(OMG, 2004). 그리고 레코드 요소 유형은 내부 논리 파일이나 외부 연계 파일 안에서 사용자가 식별 가능한 데이터 요소의 서브 그룹을 정의하는 것으로서 필수적 서브

<표 3-3> 유스케이스 기능점수(UCFP)의 데이터 기능 측정 규칙

	규칙
1	사용자가 식별할 수 있는 논리적으로 연관된 데이터 그룹 또는 제어정보로 어플리케이션의 경계를 기준으로 내부 논리 파일(ILF)과 외부 연계 파일(EIF)로 구분된다.
2	유스케이스에 의해 실현되는 기능으로 인해 어플리케이션 경계 내부에서 유지되는 수퍼 타입(Super Type) 엔티티 클래스(Entity Class) 그룹을 내부 논리 파일(ILF)로 결정한다.
3	유스케이스에 의해 실현되는 기능으로 인해 다른 어플리케이션의 경계 내부에서 유지되고 측정 대상 어플리케이션이 참조하는 수퍼 타입(Super Type) 엔티티 클래스(Entity Class) 그룹을 외부 연계 파일(EIF)로 결정한다.
4	외부 연계 파일(EIF)로 식별된 수퍼 타입(Super Type) 엔티티 클래스(Entity Class) 그룹은 측정 대상 어플리케이션에 의해 유지되어서는 안된다.

<표 3-4> 데이터 요소 유형(DET)의 식별규칙

	규칙
1	유스케이스의 단위 프로세스를 통하여 ILF 또는 EIF에서 유지 또는 검색되고, 사용자가 식별 가능하며, 반복되지 않는 유일한 클래스의 속성(Attribute)을 하나의 DET로 측정한다.
2	두 개의 어플리케이션이 같은 ILF 또는 EIF를 유지 또는 참조하되 각각 다른 DET를 유지 또는 참조한다면, 각각은 관련되는 클래스의 속성(Attribute)만을 각자의 DET로 측정한다.
3	엔티티(Entity) 클래스 사이에 연관(Association) 관계가 존재하면 연관 관계에 대한 클래스 속성(Attribute)의 수와 상관없이 이를 하나의 DET로 측정한다.

그룹과 선택적 서브 그룹의 두 가지 유형으로 분류된다. 여기에서, 필수적 서브 그룹이란 사용자가 적어도 하나 이상의 서브 그룹을 사용해야 하는 것을 말하며 선택적 서브 그룹이란 사용자가 데이터의 인스턴스(Instance)를 추가 또는 생성하는 단위 프로세스에서 서브 그룹을 사용할 수도 있고 사용하지 않을 수도 있는 것을 말한다(Garmus and Harron, 2003).

UML의 클래스 다이어그램 구성 요소 중 레코드 요소 유형의 측정에 이용할 수 있는 것으로 클래스간의 관계를 살펴볼 수 있다. 레코드 요소 유형의 서브그룹 유형 중에서 필수적 서브그룹과 일치하는 개념은 일반화(Generalization) 관계로서 이는 일반화된 사물과 보다 특수화된 사물들 사이의 관계를 표현하는 것으로 계층적인 관계를 이루는 클래스 사이에서 속성과 행동을 상속(Inheritance)한다. 레코드 요소 유형의 서브그룹 유형의 다른 하나인 선택적 서브그룹에 해당하는 관계들로는 집합 연관(Aggregation) 관계와 합성 연관(Composition) 관계가 해당되는데, 집합 연관은 부분과 전체(Part-Whole)의 관계를 나타내는 관계이며 합성 연관은 집합 연관을 확장하여 모형화 하는 것으로서 이는 모든 부분이 하나의 전체에만 속해 있어야 하며 전체가 삭제되면 부분도 따라서 삭제되어야 함을 의미한다. 연관(Association)은 클래스간의 관계를 나타내면서 동시에 클래스가 되는 것을 의미하며 이는 데이터 기능에서의 데이터 요소 유형의 측정 규칙에 따라 하나의 데이터 요소 유형으로 계산될 수 있다. 이러한 클래스의 대응 관계에 기반한 데이터 요소 유형 및 레코드 요소 유형의 측정 규칙은 <표 3-4>, <표 3-5>와 같다.

<표 3-5> 레코드 요소 유형(RET)의 식별규칙

	규칙
1	ILF나 EIF의 엔티티(Entity) 클래스 그룹 내의 클래스 각각을 하나의 RET로 측정한다.
2	특수화된 사물(Sub Type)에 대한 엔티티(Entity) 클래스가 없다면 ILF나 EIF를 하나의 RET로 측정한다.
3	상속(Inhetitance) 관계에 있는 클래스를 하나의 RET로 측정한다.

3.4.4 트랜잭션 기능의 측정 규칙

1) 트랜잭션 기능의 식별규칙

IFPUG에서는 트랜잭션 기능을 어플리케이션이 데이터를 처리하여 사용자에게 제공하는 기능이라고 정의하고 있으며, 그 유형을 외부 입력(EI, External Input), 외부 출력(EO, External Output), 외부 조회(External Inquary)의 세 가지로 규정하고 있다. 여기서 사용되는 외부라는 개념은 어플리케이션 경계의 밖을 의미하며, 측정하고자 하는 시스템과 연관을 갖는 사용자 또는 어플리케이션을 의미한다.

먼저 외부 입력(EI)은 어플리케이션 경계의 밖에서 들어오는 데이터나 제어 정보를 처리하는 단위 프로세스로서, 그 주요 의도는 하나 이상의 내부 논리 파일(ILF)을 유지하거나 시스템의 동작을 변경하는데 있다. 외부 출력(EO)는 데이터나 제어 정보를 어플리케이션 경계 밖으로 보내는 단위 프로세스로서, 주요 의도는 데이터나 제어 정보의 검색은 물론 처리 로직을 통해 사용자에게 정보를 제공하는 것이며 하나 이상의 내부 논리 파일(ILF)을 유지하거나 시스템의 동작을 변경할 수 있다. 여기서 처리 로직은 적어도 하나의 수학 공식, 계산 또는 파생 데이터를 포함해야 한다. 마지막으로 외부 조회(EQ)는 데이터나 제어 정보를 어플리케이션 경계 밖으로 내보내는 단위 프로세스로서, 주요 의도는 내부 논리 파일(ILF)이나 외부 연계 파일(EIF)로부터 데이터나 제어 정보를 검색하여 사용자에게 정보를 제공하는 것이다. 처리 로직은 수학 공식이나 계산, 파생 데이터를 포함하지 않으며, 외부 조회가 처리되는 동안 내부 논리 파일(ILF)을 유지하거나 시스템의 동작을 변경하여서는 안 된다.

객체지향 환경에서 이러한 기능점수 분석의 트랜잭션 기능에 대응하는 것은 기능점수 측정 범위에 포함되는 유스케이스라고 할 수 있다. 유스케이스는 사용자가 요구한 기능적 요구사항을 포함하고 있으며, 이러한 기능을 객체나 컴포넌트에 할당하는 것의 기반이며 시스템 경계 밖에 있는 액터와 시스템 간의 교류를 나타내는 순차(Sequence)들의 모임을 설명한다. 즉 유스케이스는 하나의 순차가 아닌 여러 개의 순차들을 통해서 이루어진다. 여기서 사용되는 순차라는 개념은 하나 이상의 트랜잭션을 포함한다고 할 수 있다. 액터는 그 성격에 따라 시작점(Initiator), 외부 서버(External server), 수신(Receiver), 수단(Facillitator)으로 분류될 수 있는데 시작점은 유스케이스를 시작시키는 역할을 하며, 외부 서버는 유스케이스 내에서 시스템에 서비스를 제공하는 역할을 수행한다. 그리고 수신은 유스케이스로부터 정보를 전달 받으며 수단은 다른 액터와 시스템간의 상호작용을 돕는다.

기능점수 계산을 위해서는 각 유스케이스 내에서 트랜잭션을 식별해 내야 한다. 이러한 작업을 위해 요구되는 자료는 유스케이스의 행동을 정형적인 방법으로 명세한 유스케이스 기술서와 이것에 의거하여 작성된 클래스 다이어그램이다.

유스케이스 모델은 복잡한 실세계를 표현하며 반복적으로 개발되는 과정을 거쳐 요구사항에 대하여 좀 더 많은 정보를 정제하는 과정을 수행하게 되며, 이러한 요구사항 분석 과정이 진행됨에 따라 문제에 대하여 좀 더 많은 이해를 할 수 있게 된다. 이러한 결과로 유스케이스 모델은 명세의 상세도를 높이면서 다양한 추상

<표 3-6> 트랜잭션 기능의 측정 규칙

	규칙
1	유스케이스 다이어그램 상에서 시작점(Initiator)을 액터로 하는 프로세스 수행이 순차적으로 진행되어서 그룹화 되는 것을 하나의 트랜잭션으로 결정한다.
2	데이터 또는 제어정보를 어플리케이션 경계 밖에서 받아들이며 새롭게 객체를 생성하거나 주어진 속성에 대응하여 자료를 입력하는 트랜잭션을 외부입력으로 결정한다.
3	객체의 변수 값이나 전달된 속성을 가지고 하나 이상의 연산을 수행하거나 비즈니스 로직을 수행하지 않으며 객체의 속성 값을 읽어오는 트랜잭션을 외부조회로 결정한다.
4	객체의 변수 값이나 전달된 속성을 가지고 하나 이상의 연산을 수행하거나 비즈니스 로직을 수행하여 새로운 값을 반환하는 트랜잭션을 외부출력으로 결정한다.

화 정도를 나타내게 된다.

요구사항을 도출한 다음으로는 제안된 요구사항을 충족시키는 유스케이스 명세서와 이에 기반하여 작성되어진 클래스 다이어그램을 분석하여 각 트랜잭션의 유형을 식별해 내야 한다. 본 단계에 사용되는 클래스 다이어그램은 전 단계에서 식별된 액터와 객체들로 구성된 것이어야 하며, 새로운 객체가 추출될 경우 새로운 객체에 대한 어플리케이션 경계의 식별이 반드시 수행되어야 한다.

2) 트랜잭션 기능의 복잡도와 기여도 식별규칙

식별된 트랜잭션 기능에 대한 기능점수 유형이 결정되면 UML의 요소들을 적용하여 데이터 요소타입(DET)과 참조파일타입(FTR, File Type Reference)의 수를 식별함으로서 트랜잭션 기능에 대한 복잡도와 기여도를 계산하게 된다.

데이터 요소타입(DET)은 사용자가 인식 가능하고 유일하며, 반복되지 않는 필드나 속성을 정의하는 것이며 이것은 객체의 속성에 대응한다고 할 수 있다(Colbert, 2001). 그리고 참조파

<표 3-7> 트랜잭션 기능의 식별 규칙

	규 칙
1	외부입력의 경우 트랜잭션 내에서 액터로부터 전달되는 메소드에 정의된 속성의 총 합을 DET로 한다.
2	외부조회의 경우 트랜잭션 내에서 액터로부터 전달되는 메소드의 속성 수와 질의결과로 반환되는 데이터 항목수의 총합을 DET로 한다.
3	외부출력의 경우 트랜잭션 내에서 액터로부터 전달되는 메소드의 속성 수와 반환되는 데이터 항목수의 총합을 DET로 한다.
4	트랜잭션의 수행 중 발생하는 다양한 메시지를 범주로 구분하여 하나의 범주를 하나의 DET로 계산한다.
5	각 트랜잭션에 관련되는 모든 객체를 각각 하나의 FTR로 계산한다.

일타입은 트랜잭션에 의해 유지되거나 읽혀지는 파일을 정의하는 것으로서 식별된 트랜잭션 내에 포함된 객체들에 대응한다. 이러한 대응관계에 기반한 측정 규칙은 다음과 같다.

3.4.5 미조정 기능점수의 계산

본 연구를 통하여 정의된 유스케이스 기능점수의 측정 규칙은 IFPUG에서 제시한 기능점수 분석절차에 UML의 요소들을 식별하여 적용한 결과로 도출되었다. 따라서 앞에서 제시된 유스케이스 기능점수 측정절차와 규칙을 적용함으로서 객체지향의 특성을 반영한 객체지향 시스템의 미조정 기능점수를 측정할 수 있다. 유스케이스 기능점수의 미조정 기능점수의 측정은 우선 기능점수 측정유형을 결정하고 측정범위와 어플리케이션 경계를 식별한 후, 데이터 기능과 트랜잭션 기능에 대한 계산 결과를 통해 측정된다.

전통적인 기능점수 분석에 있어서는 측정된 미조정 기능점수와 조정인자를 통해서 조정 기능점수를 산출하게 된다. 하지만 본 연구에서는 객체지향의 성격을 반영한 미조정 기능점수까지의 기능점수 예측을 그 목적으로 하고 있으므로 조정인자를 통한 조정 기능점수의 측정은 연구대상에서 제외하였다.

Ⅳ. 실증 분석

4.1 사례의 개요 및 특성

본 연구를 위하여 국내에서 수행된 정보시스템 개발 프로젝트를 대상으로 유스케이스 및 기능점수와 관련된 데이터들을 수집하였다. 수집된 유스케이스의 수는 총 164개로 생산 21개, 판매 70개, 물류 45개, 매장 28개, 기타 6개로 구성되었다. 유스케이스를 통해 추출된 기능의 수는 총 586개로 데이터 기능 76개와 트랜잭션 기능 510개이며 데이터 기능으로는 내부논리파일 66개, 외부연계파일 10개, 트랜잭션 기능으로는 외부입력 309개, 외부출력 47개, 외부조회 154개의 기능으로 분류하였다.

수집된 데이터들 중, 다른 데이터와 중복되거나 정보시스템 구현단계에서 수정이 이루어진 27개의 유스케이스 및 71개의 트랜잭션 기능을 제외한 데이터들을 분석대상으로 하였으며 최종적으로 143개의 유스케이스(생산 20개, 판매 55개, 물류 41개, 매장 21개, 기타 6개), 76개의 데이터 기능(내부논리파일 66개, 외부연계파일 10개), 439개의 트랜잭션 기능(외부입력 254개, 외부출력 47개, 외부조회 138개)을 분석 대상으로 선정하였다.

4.2 사례분석 방안

본 절에서는 2가지의 유스케이스 기능점수 예측기법($UCFP_1$, $UCFP_2$)과 기존에 제시되어 온 4가지 기능점수 예측기법(IFPC, EFPC, KEFPC, EFPS)을 사용하여 기능점수 사례에 대한 통계분석을 실시한다. 이를 통하여 유스케이스 기능점수 예측기법의 정확도를 다른 예측기법들과 비교분석함으로써 그 타당성을 살펴보고자 한다. 본 연구에서는 유스케이스 기능점수를 통한 사례분석을 수행하기 위하여 다음과 같은 2가지의 분석절차를 수행하였다.

4.2.1 유스케이스 기능점수 예측기법의 측정을 위한 예비조사

본 연구에서 제안된 유스케이스 기능점수 예측기법을 실무에 적용하기 위해서는 우선 유스케이스 기능점수에 대한 측정규칙과 복잡도 매트릭스, 그리고 측정 가중치가 도출되어야 한다. 유스케이스 기능점수 예측기법 중 $UCFP_2$는 선행연구의 분석을 통해 제안된 유스케이스 기능점수 측정규칙을 통하여 측정이 가능하며, 기능유형별 점수를 산정하기 위한 복잡도와 측정가중치는 IFPUG의 기능점수 측정기법에 따르므로 사례분석의 적용에 문제가 발생되지 않는다.

하지만 UCFP1은 기능점수 예측시점이 UCFP2에 비해 빠르고 측정방식 또한 IFPUG의 기능점수 측정기법을 따르지 않기 때문에 기능점수 적용사례에 대한 패턴을 분석하고 이를 통해 새로운 기능점수 복잡도와 측정 가중치의 산출이 필요하다. 다시 말해서 UCFP1의 트랜잭션 기능의 측정은 정보시스템 개발 프로젝트에 대한 빠른 예측시점으로 인하여 트랜잭션 유형별 기능점수의 산정에 반드시 필요한 데이터 요소 유형(DET)이 도출될 수 없으며, 이를 보완하기 위하여 사용한 액터와 파일 참조 유형(FTR)간의 패턴 분석을 통해 기능유형별 점수를 산정하기 위한 복잡도와 측정가중치를 도출하여야 한다. UCFP1의 데이터 기능 측정은 NESMA의 EFPC의 측정기법에 따르므로 패턴분석을 위한 예비조사에서는 제외한다.

따라서 $UCFP_1$의 트랜잭션 기능을 위한 예비조사는 사례분석의 대상인 143개 유스케이스 중에서 절반에 해당하는 72개 유스케이스를 임의로 선정하여 액터와 파일 참조 유형(FTR)간의 패턴을 분석함으로써 기능점수 복잡도와 측정 가중치를 도출한 후, 나머지 절반에 해당하는 71개 유스케이스를 통해 도출된 기능점수 복잡도와 측정 가중치에 대한 적정성을 검증한다.

4.2.2 유스케이스 기능점수 예측기법의 측정을 위한 사례분석

본 연구에서 제안된 유스케이스 기능점수($UCFP_1$, $UCFP_2$)에 대한 예측기법의 정확성을 검증하기 위하여 정보시스템 개발 프로젝트의 진행에 따른 기능점수의 예측시점을 UP를 기준으로 착수 단계와 상세 단계의 종료시점까지의 2가지로 구분하고, 각각의 예측시점에 적용 가능한 $UCFP_1$, $UCFP_2$의 2가지 방법론을 적용하였다. 또한 기존에 제시되어 온 기능점수 예측기법들과의 정확도를 비교하기 위하여 NESMA의 IFPC(Indicate Function Point Count) 및 EFPC (Estimated Function Point Count) (NESMA, 2004), ISBSG의 EPFS (Early Prediction of Function Size)(ISBSG, 2004)의 3가지 예측기법을 선정하였으며 NESMA의 EFPC에 정보통신부의 소프트웨어 사업대가 산정기준(정보통신부, 2005)의 평균 복잡도 가중치를 적용하여 국내의 현실에 맞게 수정한 KEFPC(Korea EFPC)를 추가하여 4가지의 기능점수 예측기법을 사용하였다.

본 연구에서는 기능점수 규모의 측정 대상을 객체지향 분석 절차에 따라 새로 개발되는 정보시스템으로 한정하였다. 또한 분석대상으로 선정한 기능점수 예측기법들의 정확성을 비교하기 위한 기준으로는 최종 기능점수를 사용하였으며, 여기에서의 최종 기능점수란 객체지향 기반의 정보시스템 개발이 완료된 시점에서 IFPUG의 기준에 따라 측정된 미조정 기능점수

를 의미한다.

본 연구에서 분석대상으로 선정한 6가지 기능점수 예측기법들의 정확성을 비교하기 위해 우선 평균상대오차(MMRE)를 통해 기능점수 측정항목 각각을 분석단위로 하여 각 항목별 오차율에 대한 평균을 분석한다(Conte et al., 1986; 박찬규 외, 2003; 박주석, 정기원 2004). 두 번째는 IFPUG의 미조정 기능점수와 예측된 미조정 기능점수간의 오차율 분석을 통해 각 기능점수 예측기법들에 대한 최종 기능점수 값의 예측 오차율을 분석한다.

4.3 예비 조사

$UCFP_1$의 트랜잭션 기능의 측정을 위한 예비조사는 액터와 파일 참조 유형(FTR)간의 패턴 분석을 통해 기능유형별 점수를 산정하기 위한 복잡도와 측정가중치를 도출하기 위해 수행한다. 사례분석의 대상인 143개의 유스케이스 중에서 절반에 해당하는 72개 유스케이스를 임의로 선정하여 액터와 파일 참조 유형(FTR)간의 패턴을 분석하였으며 나머지 절반에 해당하는 71개의 유스케이스를 통해 도출된 기능점수 복잡도와 측정 가중치에 대한 적정성을 평균상대오차(MMRE)를 통해 검증한다.

먼저 패턴 분석을 위해 선정된 72개 유스케이스에 대한 기술통계량을 살펴보면 다음과 같다.

<표 4-1> $UCFP_1$의 트랜잭션 기능에 대한 예비조사 기술통계량

	기능 유형	기능 점수	FTR수	객체수	오차율	DET수	액터수	오차율
EI	129	4.58	2.50	3.41	0.46	18.38	5.79	-0.67
EO	25	5.88	3.32	4.08	0.18	15.76	6.56	-0.64
EQ	69	4.55	2.72	3.16	0.30	16.29	5.62	-0.64
전체	223	1052	593	760	0.38	3889	1299	-0.66

<표 4-1>은 $UCFP_1$ 예측기법의 71개 유스케이스에 대한 트랜잭션 기능유형의 평균값을 나타내고 있다. 분석대상 유스케이스를 통해 산출된 트랜잭션 기능의 수는 223개로서 외부입력(EI) 129개, 외부출력(EO) 25개, 외부조회(EQ) 69개로 구분되었으며, IFPUG의 참조파일유형(FTR)과 $UCFP_1$ 예측기법의 객체수에 대한 오차율은 외부입력 0.46, 외부출력 0.18, 외부조회 0.30으로 나타났으며 전체적으로는 0.38의 오차율을 가지고 있는 것으로 분석되었다. 또한 IFPUG의 데이터요소유형(DET)과 $UCFP_1$ 예측기법의 액터수에 대한 오차율은 외부입력 -0.67, 외부출력 -0.64, 외부조회 -0.64로 나타났으며 전체적으로는 -0.66의 오차율을 가지고 있는 것으로 분석되었다. 이러한 결과는 $UCFP_1$ 예측기법의 객체 수는 IFPUG의 참조파일유형(FTR)보다 전체적으로 과대계상이 되고 있음을 의미하며, 액터의 수는 데이터요소유형(DET)보다 전체적으로 과소계상이 되고 있음을 의미한다.

<표 4-2> $UCFP_1$의 트랜잭션 기능유형별 복잡도 구성요소의 평균

구분	EI			EO			EQ		
복잡도	낮음	보통	높음	낮음	보통	높음	낮음	보통	높음
객체수	1.00	1.95	6.29	1.00	2.67	4.75	1.15	2.16	5.27
액터수	2.46	3.67	9.79	2.00	5.11	8.17	2.92	4.00	8.24

<표 4-2>는 $UCFP_1$의 트랜잭션 기능유형에 따른 복잡도를 구성하는 객체와 액터의 수에 대한 평균값을 나타내고 있으며, 이는 IFPUG의 최종 기능점수를 기준으로 하여 $UCFP_1$ 예측기법의 분석대상인 72개 유스케이스, 223개의 트랜잭션 기능을 기능 유형별 복잡도에 따라 분류하고 이를 분석한 것이다. 예비조사를 통해 분

석된 트랜잭션 기능유형별 복잡도에 대한 객체와 액터의 평균값은 산술평균이 가지는 통계학적인 한계로 인하여 연속적이 아닌 단편적인 값으로 표현되므로 트랜잭션 기능에 대한 복잡도의 도출에 직접적으로 사용하기에는 무리가 있다. 따라서 참조파일 유형(FTR)과 데이터요소유형(DET)에 대한 객체와 액터의 오차율을 추가로 참조하여 <표 4-3>과 같이 $UCFP_1$의 트랜잭션 기능에 대한 복잡도 매트릭스를 도출하였다.

<표 4-3> $UCFP_1$의 트랜잭션 기능에 대한 복잡도 매트릭스 및 측정 가중치

EI 복잡도 매트릭스			
객체	1-3ACT	4-7ACT	8+ACT
0-1	낮음	낮음	보통
2	낮음	보통	높음
3+	보통	높음	높음

EO/EQ 복잡도 매트릭스			
객체	1-3ACT	4-6ACT	7+ACT
0-1	낮음	낮음	보통
2-3	낮음	보통	높음
4+	보통	높음	높음

트랜잭션 기능점수 측정 가중치			
기능유형	낮음	보통	높음
외부입력(EI)	3	4	6
외부출력(EO)	4	5	7
외부조회(EQ)	3	4	6

4.4 정확도 비교 분석

4.4.1 평균상대오차 (MMRE, Mean Magnitude Relative Error)

유스케이스 기능점수 예측기법을 검증하기 위한 사례분석의 첫 번째 단계로 평균상대오차 분석을 실시하였으며, 이를 통하여 최종적으로 측정된 IFPUG의 미조정 기능점수(UFP)에 대한 각 예측기법들의 정확성을 측정하고자 하였다. IFPUG의 미조정 기능점수에 대한 분석대상 예측기법들에 대한 평균상대오차는 <표 4-4>와 같다.

<표 4-4> 분석대상 예측기법의 미조정 기능점수에 대한 평균상대오차

	데이터 기능 점수	데이터 기능 MMRE	트랜잭션 기능점수	트랜잭션 기능 MMRE
FP	610	-	2103	-
IFPC	-	-	-	-
EFPC	512.00	0.12	1803.00	0.25
EPFS	602.46	0.18	1918.43	0.29
KEFPC	535.80	0.14	1736.20	0.26
$UCFP_1$	512.00	0.12	2341.00	0.22
$UCFP_2$	630.40	0.02	2182.00	0.05

평균상대오차에 따른 분석대상 예측기법의 정확성 분석 단계에서 NESMA의 IFPC 예측기법은 도출과정의 특성상 기능점수 분석사례의 기능유형별 평균상대오차가 계산되지 않으므로 분석단계에서 제외되었다. 먼저 데이터 기능점수에 대한 분석대상 예측기법의 미조정 기능점수에 대한 평균상대오차를 살펴보면 IFPUG의 데이터 기능에 대한 미조정 기능점수는 총 610점으로 나타났으며 IFPUG의 미조정 기능점수에 대한 분석대상 예측기법의 평균상대오차는 $UCFP_2$가 0.02로 IFPUG의 미조정 기능점수(UFP)를 가장 정확하게 예측하고 있는 것으로 나타났다. 그 다음의 예측 정확도를 보이는 기법으로는 EFPC와 $UCFP_1$으로 평균상대오차가 0.12로 나타났다. 결국, 본 연구에서 제안된 유스케이스 예측기법들은 모두 정확성이 다른 예측기법에 비해 우수한 것으로 분석되었다.

다음으로 트랜잭션 기능점수에 대한 분석대상 예측기법의 미조정 기능점수에 대한 평균상

대오차를 살펴보면 IFPUG의 데이터 기능에 대한 미조정 기능점수는 총 2103점으로 나타났으며 IFPUG의 미조정 기능점수에 대한 분석대상 예측기법의 평균상대오차는 $UCFP_2$가 0.05로 IFPUG의 미조정 기능점수(UFP)를 가장 정확하게 예측하고 있는 것으로 분석되었다. 그 다음의 예측 정확도를 보이는 기법으로는 $UCFP_1$과 EFPC의 평균상대오차가 0.22와 0.25로 나타났다, 본 단계에서의 기능점수 사례분석 결과, 기능점수 분석사례의 개별적인 측정항목 각각에 대한 정확성은 $UCFP_2$, $UCFP_1$ EFPC, KEFPC, EPFS 순으로 높게 나타났다.

4.4.2 기능점수 오차율 (Function Point Relative Error Ratio)

유스케이스 기능점수 예측기법에 대한 사례분석의 마지막 단계로 IFPUG의 기능점수 측정값에 대한 예측기법의 기능점수 오차율을 측정하였다. 기능점수 오차율 측정을 통한 예측 정확성의 분석단계에서는 IFPUG의 데이터 기능과 트랜잭션 기능에 대한 미조정 기능점수를 사용하였으며 이에 대한 기능점수 예측기법별 기능점수의 오차율은 <표 4-5>와 같다.

<표 4-5> 기능점수 예측기법별 기능점수 오차율

	미조정 기능 점수	오차율 (%)
FP	2713.00	-
IFPC	2460.00	- 9.33
EFPC	2315.00	- 14.67
EPFS	2520.89	- 7.08
KEFPC	2272.00	- 16.26
$UCFP_1$	2853.00	5.16
$UCFP_2$	2803.00	3.32

먼저 본 연구에서 제안된 유스케이스 기능점수 예측기법($UCFP_1$, $UCFP_2$)들은 IFPUG의 기능점수를 다소 과대계상 하는 경향이 있으며, 기존에 제시되어 온 IFPC, EFPC, EPFS와 EFPC를 국내의 실정에 맞게 수정한 KEFPC는 모두 기능점수를 과소계상 하는 경향이 있는 것으로 나타났다. 또한 IFPUG의 최종 기능점수에 대한 기능점수 예측기법의 예측 오차율은 $UCFP_2$가 3.32%, $UCFP_1$가 5.16%의 순으로 높게 나타났으며, 기존에 제시되어 온 예측기법들 중에서는 EFPS가 -7.08%, IFPC가 -9.33%, EFPC가 -14.67%, KEFPC가 -16.26%의 순으로 예측 오차율을 나타냈다.

본 연구에서 가장 낮은 예측 정확도를 보이는 KEFPC는 NESMA의 EFPC를 국내의 현실에 맞도록 정보통신부의 소프트웨어 사업대가 산정기준에 따른 평균 복잡도 가중치를 사용하여 수정한 예측기법이지만 IFPUG의 미조정 기능점수에 대한 예측기법별 기능점수 오차율의 분석결과로는 기능점수를 가장 과소계상하고 있다. 이는 정보통신부에서 고시한 평균 복잡도 가중치가 지나치게 낮게 책정되어 있음을 의미한다고 볼 수 있다.

V. 결 론

본 연구는 최근의 정보시스템 개발 환경으로 널리 사용되고 있는 객체지향 분석기법을 기능점수 분석기법에 적용시키고, 기능점수의 예측시점을 정보시스템 개발 프로젝트의 초기로 앞당기기 위한 목적을 가지고 수행되었다.

본 연구에서 나타난 연구 결과 및 의의는 다

음과 같다. 첫째, 본 연구에서는 객체지향 분석 기법을 통한 기능점수 측정을 새로운 시각에서 접근하였다. 우선 기능점수를 위한 전통적인 측정기준은 어플리케이션과 단위 프로세스였으나 이를 객체지향의 환경에 맞추어 클래스와 유스케이스를 통한 기능점수의 측정방안을 제시하였다. 또한 트랜잭션 기능의 측정을 위한 전통적인 측정기준은 참조파일 유형 및 데이터요소 유형인데 반해 본 연구에서는 유스케이스 내의 액터와 객체의 수를 통한 트랜잭션 기능의 측정 방안을 제시하였다.

둘째, UML 및 UP의 산출물을 이용하여 정보시스템 기능점수 예측에 직접적으로 적용할 수 있는 보다 체계적이고 구체적인 객체지향 기능점수 측정 방안을 제시함으로써, 최근의 정보시스템 개발환경에서 정확한 기능점수 측정을 위한 토대를 마련하였다.

셋째, 객체지향 기반의 정보시스템 개발 프로젝트의 보다 빠른 시점에 기능점수의 예측이 가능한 기능점수 예측기법을 제시하였으며, 이를 통하여 정보시스템에 대한 정확한 자원배분을 가능하게 하고 결과적으로 정보시스템 개발 프로젝트를 성공적으로 수행할 수 있도록 가이드라인을 제시하였다.

넷째, 실제 프로젝트의 데이터를 사용한 사례분석을 통하여 본 연구에서 제안한 2가지 예측기법과 기존에 제시되어 온 4가지 기능점수 예측기법들의 정확성을 비교 분석하였는데 본 연구에서 제안한 2가지 예측기법이 훨씬 정확하게 예측하고 있는 것으로 나타났다. 따라서 본 연구의 결과는 기능점수 예측기법 사용자들이 정보시스템 개발 초기의 2가지 예측시점별로 상황에 맞는 예측기법을 선택할 수 있도록 하였다.

본 연구에는 앞에서 논의한 여러 가지의 연구 의의가 있지만 다수의 한계점도 동시에 존재한다. 본 연구에서는 정보시스템 개발 프로젝트의 특성상 정보시스템 개발 정보를 담고 있는 유스케이스와 정보시스템 개발노력과 소요 비용의 정보를 담고 있는 기능점수 관련 자료의 수집에 많은 어려움이 발생하였으며 제한된 자료수집으로 인해 본 연구에서 제안한 유스케이스 기능점수의 일반화 문제가 제기될 수 있다. 예를 들어 유스케이스 기능점수의 측정기준으로 사용된 유스케이스와 클래스 다이어그램과 같은 객체지향 분석기법의 산출물은 작성의 명세화 수준과 정확도 등에 의해 기능점수 예측의 정확도가 영향을 받을 수 있다. 따라서 객체지향 분석기법의 산출물에 대한 정확한 작성기준과 명세화의 수준을 실무의 요구에 맞추어 제시한다면 후속 연구에 좋은 기초 자료로 활용될 수 있을 것이다. 또한 본 연구에서는 복잡한 계산과 처리가 중심인 과학기술 데이터와 금융권 데이터의 처리와 같이 데이터의 입력 수가 적고 처리과정이 복잡한 프로세스에 대한 기능점수 측정의 고려가 이루어지지 않았다. 이러한 문제점은 현재까지 기능점수가 가지고 있는 근본적인 문제점으로 다양한 분야의 프로세스와 시스템의 특성을 포함하는 기능점수 관련 연구도 좋은 연구 주제가 될 수 있을 것으로 판단된다.

참고문헌

박주석, 정기원, "소프트웨어 개발 비용을 추정하기 위한 사용사례 점수 기반 모델,"

정보처리학회논문지, 제11-D권, 제1호, 2004년 2월, pp. 163-172.

박찬규, 신수정, 이현옥, "국내 소프트웨어 개발 사업에 적합한 기능점수규모 예측방법에 관한 연구," 경영과학 제20권, 제2호, 2003년 11월, pp.179-196.

유철중, 정소영, "요구사항 기술서로부터 유스케이스 다이어그램의 추출기법," 정보처리학회 논문지, 제9-D권, 제4호, 2002, pp. 639-650.

정보통신부, 소프트웨어 사업대가 기준, 제2005-22호, 2005.

정승렬, 임좌상, "객체지향 시스템 개발에서의 유스케이스 활용성에 관한 연구," Journal of Information Technology Applications & Management, 제12권, 제1호, 2004. pp. 21-34.

Abran, A. and P. N. Robillard, "Function Points Analysis: An Empirical Study of its Measurement Processes," *IEEE Transactions on Software Engineering*, vol. 22, Issue. 12, Dec. 1996, pp. 895-910.

Albrecht, A. J., "Measuring Application Development Productivity," Proceedings of Joint Share/Guide/IBM Application Development Symposium, Oct. 1979, pp. 83-92.

Ambler, S. W., The Unified Modeling Language v1.1 and Beyond: The Techniques of Object-Oriented Modeling - An AmbySoft White Paper, SIGS Books, Cambridge University Press, 1998.

Colbert, E., Introduction to the Unified Modeling Language(UML), Absolute Software, Sapphire Dr, Carlsbad, CA, 2001.

Colbert, E., An Overview of Object-Oriented Software Development with the Unified Modeling Language, Absolute Software, Sapphire Dr, Carlsbad, CA, 1999.

Conte S. D. et al., Software Engineering Metrics and Models, Benjamin/ Cummings Publishing Company, 1986.

Dolado, J. J., "A Study of the Relationship among Albrecht and Mark II Function Points, Lines of Code 4GL and Effort," *Journal of Systems Software*, vol. 37, 1997, pp. 161-173.

Emrick, R. D., "In Search of a Better Metric for Measuring Productivity of Application Development," IFPUG, 1987.

Furey, S., "Why we should use Function Points," *IEEE Software*, vol. 14, Issue 2, 1997, pp. 28-29.

Garmus, D. and D. Harron, Measuring The Software Process - A Practical Guide to Functional Measurements, Prentice Hall, Pearson Education, 2003.

Garmus, D. and D. Harron, Function Point Analysis - Measuring Practices for Success Software Projects, Addison-Wesley, Pearson Education, 2001.

IFPUG, Function Point Count Practice Manual, Release 4.2, International Function Point Users Group, 2004.

ISBSG, Estimating Function Point Size, International Software Benchmarking Standard Group, 2004, http://www.isbsg.org/html/fpsize.html

Jacobson, I., Christerson, M. and P. Johnson, Object-Oriented Software Engineering: A Use Case Driven Approach, Addison-Wesley, 1992.

Jones, T. C., "A Short history of Function Points and Feature Points," Software Productivity Research Inc., 1988.

Kemerer, C. F. and B. S. Porter, "Improving the reliability of function point measurement: an empirical study software engineering," *IEEE Transactions on Software Engineering*, vol. 18, Issue. 11, Nov. 1992, pp. 1011-1024.

Kusumoto, S., Inoue, K., Kasimoto, T., Suzuki, A., Yuura, K. and M. Tsuda., "Function Point Measurement for Object-Oriented Requirements Specification," Proceedings of International Computer Software and Applications Conference, 2000, pp. 543-548.

Larman, C., *Applying UML and Patterns: An Introduction to Object Oriented Analysis and Design and thr Unified Process*, 2nd Edition, Prentice Hall, 2002.

Lokan, C. J., "An Empirical Analysis of Function Point Adjustment Factors," *Information and Software Technology*, Vol. 42, 2000, pp. 649-660.

Low, G.C. and D. R. Jeffery, "Function Points in the estimation and evaluation of the software process," *IEEE Transactions on Software Engineering*, Vol. 16, No. 1, 1990. pp. 64-71.

Matson, J. E., Barrett, B. E. and J. M. Mellichamp, "Software Development cost estimation using function points," *IEEE Transaction on Software Engineering*, Vol. 20, No. 4, 1994, pp. 275-287.

NESMA, Early Function Point Counting, Netherlands Software Metrics Association, 2004, http://nesma.nl/en

Nurcan, S., Grosz, G. and C. Souveyet, "*Describing Business Processes with a Guided Use Case Approach*," Lecture Notes in Computer Science 1413, Pernici, C., Thanos, Springer, Pisa, Italy, June 1998.

OMG, UML Superstructure 2.0 Draft Adopted Specification, Object Management Group, Oct. 2004.

Orr, G. and Reeves, T. E., "Function Point Counting: One Program's Experience," *Journal of Systems and Software*, Vol. 53, No. 3, 2000, pp. 239-244.

Park, R. E., Goethert, W. B. and W.A. Florac, Goal Driven Software Measurement a

Guidebook, Software Engineering Institute, Carnegie Mellon University, August 1996.

Pressman, R. S., *Software Engineering: A Practitioner's Approach*, 4th Edition, McGraw-Hill, 1997.

Putnam, L. and W. Myers, Measures for excellence, Yourdon Press, 1992.

Ribu, K., "Estimating Object Oriented Software Projects with Use Cases," University of Oslo Department of Informatics, Master of Science Thesis, 2001.

Schach, S. R., *Introduction to Object-Oriented Analysis and Design with UML and the Unified Process*, McGraw Hill, 2004.

Shinji, *et al.*, "Function Point Measurement from Java Programs," *International Conference on Software Engineering*, May 2002, pp. 576-582.

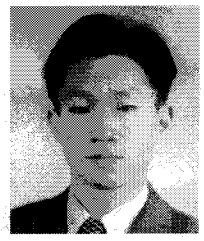
Symons, C., COSMIC-FFP Measurement Manual ver.2.2 - The COSMIC Implementation Guide for ISO/IEC 19761:2003, COSMIC, Jan 2003.

Uemura, T., Kusumoto, S. and K. Inoue, "Function Point Analysis for Design Specifications Based on the Unified Modeling Language," *Journal of Software Maintenance and Evaluation*, Vol. 13, No. 4, 2001, pp. 223-243.

정승렬(Seung-Ryul Jeong)

정승렬은 미국 위스컨신 대학에서 경영정보학 석사를, 그리고 사우스 케롤라이나 대학에서 경영정보학 박사를 취득하였다. 현재 국민대학교 비즈니스 IT학부 교수로 재직 중인 그는 프로세스 관리, ERP, 정보자원관리, 정보시스템 감리, 시스템 구현 등의 주제와 관련하여 국내외 유명 저널에 다수의 논문을 발표하였다.

이석준(Suk-Joon Lee)



이석준은 국민대학교 대학원 정보관리학과에서 경영정보시스템 전공으로 경영학 석사와 정보관리학 박사를 취득하였다. 현재 SK C&C에 재직중이며 주요 관심분야는 기업정보화전략, 변화관리, IT산업정책, e-비즈니스 등이다.

<Abstract>

Estimating the Function Point for the Object Oriented Information Systems

Seung-Ryul Jeong · Suk-Joon Lee

The purpose of this study is to present a new function point estimation approach for the Object-Oriented information systems. In order to fulfill this purpose, we first review the literature on Function Point Analysis of IFPUG, Unified Modeling Language, and Unified Process. Then, we derive a method and rules for estimating Function Points based on Use Cases and Class Diagrams. To analyze the appropriateness of the proposed approach, we conduct the empirical testing. 143 use cases are collected from production, marketing, distribution, sales, and other areas from the Object-Oriented systems development projects. We compare our new approach with the existing methods that are usually used for traditional systems development projects. The results show that our proposed approach is more appropriate for the Object-Oriented environment.

Keywords: Function Point, Object Oriented Information Systems, Unified Modeling Language

* 이 논문은 2006년 12월 20일 접수하여 1차 수정을 거쳐 2007년 1월 7일 게재 확정되었습니다.