기능점수와 프로젝트 개발요소간 상호관계에 관한 연구

이원호

경복대학 인터넷정보과

A Study on Relationship between Function Points and Development Factors in Projects

Lee, Wonho
Kyungbok College
E-mail : whlee@kyungbok.ac.kr

요 약

일반산업 분야의 프로젝트에 비해 소프트웨어 개발 분야에서의 프로젝트는 그 자체 특성상 가시적이고 계량적으로 평가하기 어려웠고 이로 인해 그 동안 국내 업계 및 정부기관에서는 프로젝트 규모 및 비용 산정에 있어서 많은 어려움이 있었다. 다행히도 최근 국제적으로 소프트웨어 프로젝트 측정기준으로 자리잡고 있는 기능점수에 관해서 국내 학계 및 업계에서도 관심을 가지게 되었고 관련된 연구결과도 계속 나오고 있다. 또한, 정부기관을 비롯하여 업계에서도 이러한 연구 결과를 기반으로 소프트웨어 개발비, 유지보수 및 재 개발비의 산정 기준으로 기능점수를 활용하려는 시도가 이루어지고 있다. 이러한 기능점수의 산정에 있어서 프로젝트의 주요 개발 요소인 하드웨어 플랫폼, 개발언어 유형, 대상업무 유형, 개발 납기 등에 따라 보정계수가 다르게 적용되고 있다. 그러나 아직 국내에서는 기능점수로 프로젝트 비용을 산정하기 위한 공감대 및 신뢰도에 대한 확산이 안된 상태이고 이에 따른 분쟁의 소지도 여전히 가지고 있다. 이러한 시점에서 본 논문은 ISBSG(International Software Benchmarking Standards Group)에서 제공하는 그 동안 축적된 해외 프로젝트 개발 결과 데이터를 기초로 기능점수와 프로젝트 개발에 투입되는 주요 개발요소간 관련성 및 프로젝트 개발 요소간의 상관관계를 분석 및 검증하여, 프로젝트 내에서의 기능점수와 관련된 개발요소간의 관련 정도에 대한 통계적 근거를 제시하여 향후 기능점수와 관련해서 이와 유사한 국내 프로젝트 수행 시 벤치마킹 자료로 활용하는데 그 목적이 있다.

1. 서론

1.1 연구 배경 및 목적

소프트웨어 규모 및 비용산정에 많이 사용되어지고 있는 기능점수(Function Points)는 1979년에 Albrecht[6]에 의해 기능점수 모형이 처음 제시된 이후, 국제적으로 소프트웨어 개발 프로젝트와 관련해서 현재 학계와 산업계에서 많이 사용되어지고 있는 IFPUG (International Function Point User Group) 기능점수 모형까지 꾸준히 발전되어 왔다. 그 외 Symons에 의해 제안된 Mark II 기능점수 모형, Jones의 특성 점수 모형, COSDES 모형 등이 있으나[4], 본 연구에서는 일반적으로 많이 사용되어지고 있는 IFPUG 기능점수를 기준으

로, ISBSG (International Software Benchmarking Standards Group)에서 제공하는 소프트웨어 개발 프로젝트관련 축적 데이터(Data Repository)를 가지고 분석 연구하고자 한다. ISBSG[15]는 1996년에 호주에 본부를 두고 창설된 조직으로, 현재 국제적으로 소프트웨어 개발 프로젝트 관련 데이터의 계속적인 축적 및 공급, 프로젝트 벤치마킹용 소프트웨어인 벤츄리(Venturi)의 개발 및 공급, 프로젝트 벤치마킹 관련서적 발간 등 활발히 활동하고 있다. 본 연구의 목적은 소프트웨어 개발 프로젝트의 측정지수로 널리 사용되어지고 있는 IFPUG 방식의 기능점수와 프로젝트의 주요 개발 요소들인 프로그램언어의 종류, 개발 방법론의 사용유무, 개발 형태의 종류, 개발 플랫폼의 종류, 개발기간, 결함 수, 노력지수, 팀 크기 등이 상호 영향력을 어느 정도 미치는지를 ISBSG가 제공하고 있는 최근 자료(Data Disk Release 6.0)를 통해 검증 및 확인하여, 향후 이와 유사한 개발환경에 있는 프로젝트 관리자 및 개발자에게 사전에 유용한 정보를 제공한다는데 그 목적이 있다.

1.2 논문구성

본 논문의 구성은 다음과 같다. 다음 제 2장에서는 기능점수의 개념 및 산출방식, 기능점수와 소프트웨어 생산성과의 관계, 프로젝트의 주요 개발 요소와 생산성과의 관계에 대한 이론적 고찰을 통해서 가설설정의 근거를 제시한다. 제 3장에서는 앞서 제 2장에서 제시된 이론적 내용을 기반으로 설정된 가설을 정리하였고, 다시 설정된 가설을 중심으로 연구모형을 만들어 제시한다. 제 4장에서는 연구모형에서 나타난 연구변수를 정의한 후, 가설을 검증하기 위한 자료 수집 및 추출 과정, 자료의 특징 및 분석방법에 대해 차례로 기술한다. 제 5장에서는 통계결과를 통해 가설을 검증함으로서 획득할 수 있는 시사점 및 의미에 대해 논의한다. 끝으로 결론에서는 연구결과를 다시 한번 요약정리하고, 본 연구의 기여도, 한계점, 향후 연구과제에 대해 제시한다.

2. 문헌고찰

2.1 기능점수 개념 및 산출 방식

미국 IBM의 A.J. Albrecht[6][7]는 소프트웨어 시스템의 기능을 외부입력, 외부출력, 내부논리파일, 외부인터페이스파일, 외부조회 등 다섯 가지 수행 기능으로 분류한 후, 각각의 기능 수를 산정한 후 전체 기능의 합으로 산출하는 기능점수의 개념을 제안하였다. Albrecht[6][7]의 기능점수는 기존의 소프트웨어 규모 척도로 오랜 동안 사용해 왔던 코드 수(LOC : Line of Code)의 문제점을 개선시킨 평가 방법으로 계속 주목을 받아왔고, 현재에는 Albrecht의 기능점수 모형에서 발전된 IFPUG 기능점수 모형이 많이 사용되어지고 있다.[12] IFPUG 기능점수 산출 방식[14]은 Albrecht의 기능점수 산출의 근간이 되는 다섯 개의 기능은 그대로 유지하면서 각 기능별 복잡도와 값 조정인자(VAF : Value Adjusted Factor)를 추가하여 기능점수의 정확성을 더욱 향상시킨 방법이다. ISBSG가 제공하고 있는 프로젝트 개발 관련 자료 대부분은 이와 같은 IFPUG 방식으로 산출된 기능점수로 되어 있고 소수의 프로젝트만 Mark II 방식의 기능점수로 되어있다. 따라서 본 연구에서는 풍부하고 객관적인 프로젝트별 비교분석을 위해 IFPUG 기능점수로 제공되어지고 있는 프로젝트만 추출하여 연구대상에 포함시켰다.

2.2 기능점수와 생산성과의 관계

일반적으로 생산성은 투입량에 비해 산출되는 양이 많으면 높다고 할 수가 있다. 소프트웨어 분야에서도 이런 생산성 개념이 도입되어 사용되어져 왔다.[3] 앞서 연구된 내용을 정리하면 소프트웨어의 생산성 측정을 위해 사용되어지는 공통적 요소는 기능점수(FP), 코드 수(LOC), 문서분량 등으로 압축되어진다.[7][20][27] 그 중 기능점수는 코드 수 측정 방법의 문제점을 개선시킨 발전된 형태의 측정 모형이고 현재 산업체와 학계에서도 객관적인 소프트웨어 생산성 측정 수단으로 가장 인정받고 있다.[12][20][24] 또한, 기능점수는Albrecht[6][7]를 중심으로 Behrens[10], Low & Jeffery[22],Kemerer[21] 등 여러 학자들에 의해서도 소프트웨어 생산성 측정 방법으로서 타당성이 증명되었다. 기능점수는 소프트웨어 개발에서의 생산성뿐만 아니라 소프트웨어 유지보수에서의 생산성

[9][10][23][25]에서도 중요한 측정 수단으로 사용되어지고 있다. 한편, 기능점수가 높다고 하면 구현된 소프트웨어의 생산성이 높다고는 할 수가 있으나 결코 품질을 보장하는 것은 아니다. 그렇기 때문에 산출된 분량이 동일하다 해도 품질의 차이에 의해 소프트웨어의 가치가 달라지게 된다.[3] 따라서, 본 연구에서는 구현된 소프트웨어의 품질이 일정 수준이라는 전제하에, 기능점수를 프로젝트 생산성의 간접 측정지표로 간주하고 프로젝트의 주요개발 요소들은 프로젝트 생산성과 관련 있는 투입 요소로 가정하여 기능점수와 프로젝트 주요 투입 요소와의 관계를 연구하도록 한다.

2.3 주요 개발요소와 생산성과의 관계

Sawyer & Guinan[26]은 소프트웨어 개발 프로젝트의 생산성과 팀 성과에 있어서 현재 널리 사용되고있는 RAD(Rapid Application Development), JAD(Joint Application Development), OOD(Object Oriented Development) 등의 개발 방법론과 CASE(Computer Aided Software Engineering)와 같은 소프트웨어 개발도구도 중요하게 영향을 주지만, 팀을 구성하고 있는 인적 자원도 중요하게 작용되어 진다는 사실을 실증적 실험을 통해 보여주었다. 그리고 프로그래밍 언어가 어떻게 소프트웨어 생산성에 영향을 미치는 지는 아직 까지 논란의 대상이 되고 있으며,[19] 팀을 구성하고 있는 개별 요원들이 생산성이 높은 프로그램 언어를 원활히 사용함으로서 팀의 전체적인 생산성을 높일 수 있다는 사실은 계속 연구의 대상이 되고 있다.[26] Jones & Capers[17]는 새로 개발하는 신규개발과 기존 시스템을 강화하는 보강개발의 두 가지 개발형태에서 기업은 결국 비용 대 이익 측면의 생산성 관점에서 개발형태를 결정하게 된다고 주장하는 등 지금까지 주로 비용 대 이익 측면에서 연구가 많이 진행되어 왔다. 한편, 개발플랫폼에서도 전통적인 mainframe 보다는 PC를 기본으로 하는 클라이언트/서버에서의 개발비용이 적게 소요된다는 다운사이징의 사례결과[29] 등 주로 비용관점에서 연구가 많이 되어 왔다. Tarek et al.[28]은 프로젝트에서 비용 지향적인 그룹과 품질 지향적인 그룹으로 나누어 실험을 하였는데, 그 결과 개발기간은 비용 지향적인 그룹이 빨리 완성되었고 결함 수에서는 품질 지향적인 그룹에서 적게 검출되었다. 빠른 개발이 비용절감으로 이어져 생산성 향상을 가져오지만, 결함 수는 많이 발생할 수 있다는 위험을 가지고 있다는 점에 유의해야 한다. 그렇다고 개발기간이 길다고 해서 결함 수가 줄어진다든지, 반대로 개발기간이 짧다고 해서 결함 수가 늘어난다는 근거는 없다. Dana & Brent[12]의 연구 실험결과에 의하면 결함 수는 프로그램 크기가 커지면 그 만큼 결함 수도 증가하게 됨으로 코드 수(LOC) 측정 방법보다는 기능성(Functionality)을 중시하는 기능점수 당 결함 수 측정이 보다 객관적이라는 사실과 기능점수가 높으면 결함 수도 함께 줄어든다는 것이다. Bahram Adrangi & Warren Harrison[8]이 제시한 노력산정모델에 의하면 노력지수는 프로그램 크기(LOC)와 상관관계가 있고 프로그래밍 언어에 따라 차이가 나게 되는데, 일단 프로그래밍 언어와 프로그램 크기가 결정되어지면 프로젝트의 노력지수를 예측할 수 있게 된다고 하였다. Bahram Adrangi & Warren Harrison[8]의 노력산정모델에서는 노력지수와 LOC와의 상관관계만 나타낼 뿐 기능점수와의 관계는 포함하고 있지 않지만, 권기태 & 우치수[1]의 기능점수 규모산정 기법을 지원하는 비용산정 모델인 SC2 에서는 기능점수와의 관계를 반영하고 있다. Jeffery, D.R[16]에 의하면 프로젝트의 크기를 결정하는 하나의 요소인 팀 크기에서 팀원들의 많은 경력, 높은 기술수준, 적극적인 태도의 보유정도가 프로젝트의 생산성에 영향을 미친다고 하였다. 또 한편, Fried, Louis[13]에 의하면 팀 크기가 줄어들면 원활한 의사전달 및 신속한 의사결정 등 개인 당 생산성이 향상될 수 있고, 팀 크기가 큰 경우에는 팀원들의 특정 역할 분담 등을 통해 생산성을 향상시킬 수 있다고 하였다.

3. 가설설정 및 연구모형

3.1 가설 설정

앞서 제 2장에서 논의된 내용을 기초로 설정한 가설들을 정리하면 다음과 같다.

가설1 : 개발 플랫폼에 따라 기능점수에 영향을 미치는 정도가 다를 것이다.

가설2 : 개발 플랫폼별 프로젝트 개발 시

현업에서 자주 사용되고 있는 언어인 VisualBasic과 PowerBuilder 에 따라 기능점수에 영향을 미치는 정도가 다를 것이다.

가설3 : 기업의 기존 시스템을 대체하여 새로 개발하는 신규 개발 과 기존 시스템을 강화시키는 보강개발간 개발방법에 따라 개발 방법론의 사용 유무가 기능점수에 영향을 미치는 정도가 다를 것으로 예측되나, 만약 그렇치 않다면 개발 방법론의 사용유무가 기능점수 외에 최소한 한가지 이상의 다른 개발요소에는 영향을 미칠 것이다.

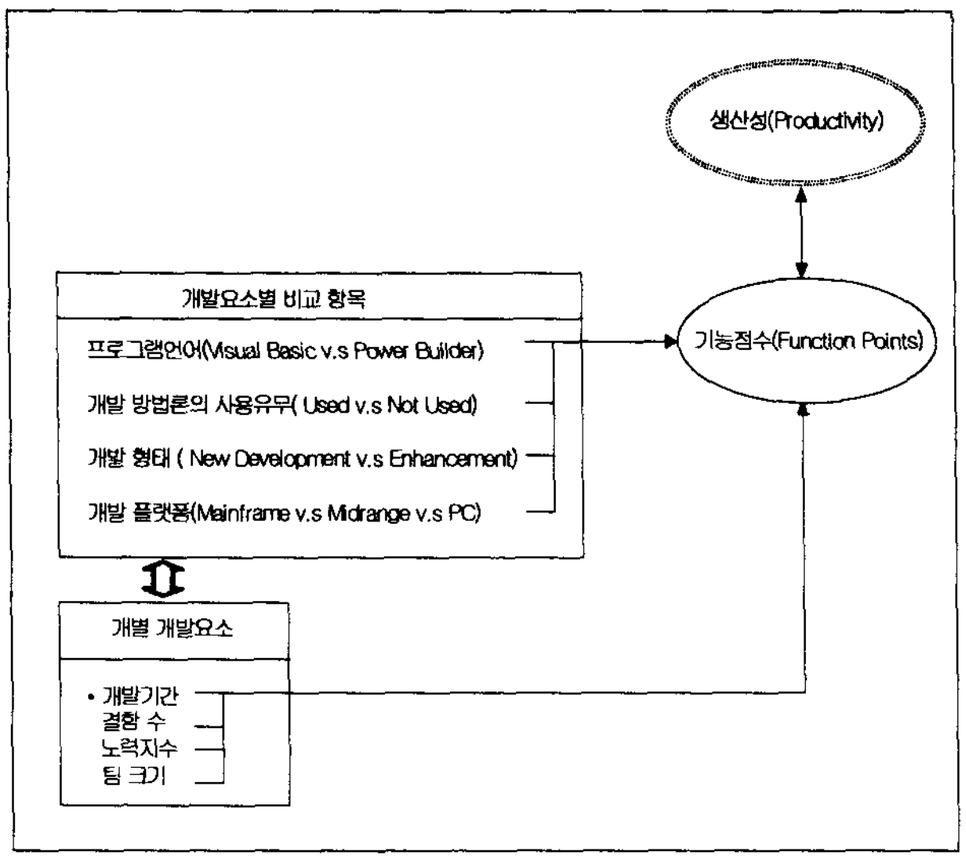
가설4 : 신규개발과 보강개발간 개발방법에 따라 기능점수 외에 노력지수, 개발기간, 팀크기 등 의 개발요소에 미치는 정도의 차이가 있을 것이다.

가설5 : 프로젝트 개발과정에서 산출되는 개발기간, 결함 수, 노력지수, 팀 크기와 같은 개발 요소와 기능점수 간에 상호 상관관계가 있을 것이다.

3.2 연구모형

앞서 설정한 가설들을 기초로 프로젝트에서의 기능점수와 주요프로젝트간의 관계를 연구모형으로 만들어 나타내면 <그림 1>과 같다.

<그림 1> 기능점수와 프로젝트 주요 개발요소간의 관계



4. 연구설계

4.1 연구 변수의 정의

4.1.1 개발요소별 비교항목

① 프로그램 언어 : 프로젝트 개발에 주로 사용된 언어로 VisualBasic, PowerBuilder 외에 PL/1, Natural, Cobol, C등의 값들이 들어 있다. 본 연구에서는 VisualBasic과 PowerBuilder만 선별 추출하여 관찰대상에 포함시킨다.

② 개발 방법론의 사용 유무 : 프로젝트 개발에 있어서 개발 방법론이 사용되었는지 또는 사용이 안되었는지의 두 가지 종류의 값으로 나타내어진다.

③ 개발형태 : 프로젝트 개발 형태는 기존의 시스템을 강화시키는 보강개발과 새로 개발하는 신규개발의 두 가지 종류 값으로 나타내어진다.

④ 개발 플랫폼 : 프로젝트 개발에 사용되어지는 개발플랫폼으로, Mainframe, MidRange, PC로 분류된다.

4.1.2 개별 개발요소

① 개발기간 : 프로젝트 개발에 소요된 총 기간을 의미하며, 월 단위로 산출되었다.

② 결함 수 : 시스템 개발 후 첫 한 달 사용 후에 보고된 결함 수이다.

③ 노력지수 : 프로젝트 개발에 소요된 전체적인 노력으로, 시간 단위로 산출되었다.

④ 팀 크기 : 프로젝트 개발에 있어서 절정 시기에 투입된 최대 인원수로 산출되었다.

4.1.3 기능점수

값 조정인자에 의해 조정된 기능점수로 IFPUG 방식에 의해 산출된 값만 추출하여 연구대상에 포함시킨다.

4.2 모집단 자료의 특징

본 연구 표본자료의 원천이 되는 모집단 자료는 ISBSG[15]에서 제공하고 있는 자료(Data Repository Release 6)로서 호주, 미국 등 세계 20여 국가의 총 791개 프로젝트에서 수집되어 축적되어온 자료들이다.

4.3 표본자료 추출방법

본 연구에서는 모집단 자료에서 앞서 설정한 연구가설들을 검증하기 위해 필요한 8개의 개발요소와 IFPUG 기능점수 항목만을 선택한 후 여기에 해당하는 자료들만 모집단 자료에서 별도로 추출하여 표본자료에 포함시켰고, 이 과정에서 모집단 정규분포에서 벗어나는 값들 또는 관련이 없고, 적용할 수 없거나 알려지지 않은 값들은 표본자료에서 제외시켰다.

4.4 표본자료 분석방법

본 연구에서는 SPSS 통계패키지를 이용하여 비교대상이 2개의 집단변수일 경우(가설2, 가설3, 가설4)는 T-검정을 활용했고, 비교대상이 3개의 집단변수일 경우(가설1)는 ANOVA-검정을 활용했다. 또한 가설5와 같이 여러 개의 개발요소간의 상호관계를 동시에 분석하기 위해서 피어슨 상관관계 분석도구를 활용했다.

5. 통계결과 및 논의

통계결과를 요약하면 <표 1>과 같다.

<표 1> 가설에 대한 통계결과 요약

가설	통계 결과
가설1 (채택)	• MainFrame vs MidRange 플랫폼의 기능점수의 차이가 가장 크게 나타남 (MidRange의 기능점수 > MainFrame의 기능점수) • MainFrame vs PC, MidRange vs PC 의 차이는 나타나지 않음
가설2 (기각)	• MidRange와 PC 플랫폼에서 사용된 PowerBuilder와 VisualBasic 간 기능점수에 미치는 차이가 없음
가설3 (채택)	• 신규개발, 보강개발 모두에서 방법론 사용유무가 기능점수에 미치는 차이가 없음 • 신규개발, 보강개발 모두에서 방법론 사용유무가 개발기간에 미치는 차이가 있음 (보강개발보다 신규개발에서 방법론 사용 시 개발기간 차이가 더 길게 나타남)
가설4 (채택)	• 개발방법(신규개발 vs 보강개발)에 따라 팀 크기를 제외한 기능점수, 노력지수, 개발기간에 미치는 차이가 있음 (신규개발이 보강개발보다 기능점수, 노력지수, 개발 기간 이 크게 나타남)
가설5 (채택)	• 개발기간과 노력지수가 기능점수와 강한 양적 상관관계 • 팀 크기와 노력지수가 강한 양적 상관관계

가설1의 결과는 과거의 많은 기업들이 다운사이징의 방법으로 MainFrame에서 MidRange 플랫폼으로 전환하거나 클라이언트/서버 또는 워크스테이션 환경을 선호하는 하나의 이유로 해석할 수가 있다. 또한 프로젝트 신규개발 시 어떠한 플랫폼을 선택해야하는 지 하나의 간접지표로서도 의미가 있다고 볼 수 있다. 가설2의 통계적 결과는 최근 현업에서 많이 사용되어지고 있는 플랫폼인 MidRange와 PC에서 PowerBuilder와 VisualBasic 간 기능점수에 미치는 차이가 없다라는 의미로 해석된다. 만약, 이 두 언어간 기능점수에 미치는 정도의 차이가 어느 한쪽으로 치우쳐 결과가 나왔다면, 두 언어를 공급하는 벤더들 및 이미 이 두 언어로 개발하여 운영하고있는 사용자들에게는 논란의 소지가 될 것으로 예측된다. 가설3의 결과는 신규개발과 보강개발에서 각각 방법론의 사용유무가 기능점수에 영향을 미칠 것이라는 처음 예상과는 달리, 기능점수가 아닌 개발기간에 영향을 미친다는 결과를 보여주었다. 통계결과 신규개발 시 방법론을 사용하면 평균 10.25 개월, 방법론을 사용하지 않으면 평균 7.66 개월 개발기간이 걸렸다는 결과를 보여주고 있으며, 이것은 기업에서 신규개발 시 방법론을 사용할 것인가 그렇지 않을 지를 정해진 개발기간 내에서 결정해야할 때 참고할 수 있다고 본다. 가설4의 결과는 신규개발이 보강개발보다 노력지수 및 개발기간 측면에서 많이 소요될 것이라는 막연한 추측에서 구체적으로 그 크기의 차이를 실제 프로젝트 수행 결과를 통해 보여주고 있다는 데에 그 의미가 있다. 기능점수는 가설5의 상관관계 결과에서도 나타난 것과 같이 노력지수와 개발기간과 강한 양적인 관계가 있고, 이로 인해 가설4의 결과에서도 신규개발 시 노력지수와 개발기간과 기능점수도 함께 증가된다 라는 의미로 해석할 수 있겠다. 한편, 가설4의 결과는 신규개발과 보강개발간의 선택이 팀 크기와는 무관하다는 것을 보여줌으로서, 일반적으로 신규개발에 많은 인력이 투입되어야 한다는 선입감을 제거해 주는 결과로 그 의미가 있다. 가설5의 결과는 개발기간이 길어질수록 그리고 노력지수가 많이 투입될수록 기능점수가 커질 것이라는 예측을 통계적 결과를 통해서 보여주고 있다는데 의미가 있다. 그리고 결함수가 많으면 일반적으로 기능점수가 감

소할 것이라고 생각할 수도 있지만 가설5의 결과를 통해서 보면 결함수가 많아도 기능점수는 증가할 수도 있다는 것이다. 또한 가설5의 결과에서 팀 크기와 노력지수가 강한 상관관계가 있다는 것을 보여주는 것은 팀을 구성하고 있는 인력이 많을수록 그만큼 노력지수가 증가한다는 것으로 해석되어질 수 있지만 그 것이 바로 기능점수의 직접적인 증가로 이어진다는 사실은 아니다.

6. 결론

본 연구는 프로젝트의 산출물 중의 하나인 기능점수를 중심으로 프로젝트 개발과정에서 산출되어 질 수 있는 그 외 여러 가지의 개발요소와의 관계를 세계 각 국에서 수행되었던 프로젝트 결과물들을 대상으로 다양한 시각을 가지고 추적해 보았다. 본 연구의 결과는 현업에서 프로젝트 수행 시 여러 가지 개발요소들 중에서 일부를 선택해야 하는 어려운 문제 등에 직면했을 때 앞서 수행되었던 여러 프로젝트 결과를 통해서 좀더 생산성을 높일 수 있는 방향으로 의사결정을 할 수 있도록 통계적 근거를 제시해 주고 있다는 데 그 의의가 있다. 그러나 본 연구에서 통계자료로 사용한 ISBSG의 축적된 데이터는 하나의 산업에 속한 기업들의 프로젝트가 아닌 다양한 여러 산업으로 섞여있는 상태로, 동일한 하나의 산업에서의 프로젝트들 간의 결과는 본 연구 결과와 다르게 나올 수도 있다는 것이다. 또한, 수집된 프로젝트의 개발시점, 개발대상 업무의 난이도, 그리고 해당 국가의 문화, 경제적 환경 차이를 무시하고 데이터를 혼합하여 단일한 기준으로 처리하는 과정에서 결과가 왜곡되어질 수 있는 가능성도 배제할 수는 없다. 이와 함께 본 연구에서 사용한 데이터에는 불행히도 국내에서 수행되었던 프로젝트 결과가 없었다는 데도 우리나라 현실과는 차이가 있을 수도 있다. 그러나 이런 모든 한계점을 가지고 있으면서도 본 연구의 결과는 절대적인 기준은 아니지만은 프로젝트 개발 관련하여 하나의 가이드 라인을 제공하고 있다. 향후 연구과제로는 ISBSG와 같이 해외에서 수집된 자료가 아닌 순수하게 국내 프로젝트 수행결과의 자료를 대상으로 통계결과를 산출한 뒤 본 연구 결과와 비교하여 그 결과가 같거나 유사한지, 아니면 전혀 다른 결과가 나오는지 검증할 필요가 있다. 또한, 본 연구의 주요 관찰 대상인 되었던 개발요소 외에 기능점수에 영향을 미칠 수 있는 개발요소를 추가로 발굴하여 그 범위를 확장하면 현업에서의 다양한 요구에 충족할 수 있는 결과를 축적할 수 있을 것으로 기대한다.

[참고문헌]

[1] 권기태, 우치수, “기능점수 규모산정을 지원하는 비용산정 모델”, 정보과학회논문지 제24권 제11호, 1997. 11

[2] 김현수, “기능점수를 이용한 소프트웨어 규모 및 비용산정 방안에 관한 연구”, 경영과학 제14권 제1호, 1997. 5

[3] 김현수, "프로젝트 측정 및 관리시스템 개발연구", 쌍용정보통신, 1998

[4] 안연식, “소프트웨어 규모예측을 위한 기능점수모형의 개선“, 한국정보처리학회 논문지 제4권 제4호, 1997. 4

[5] 최은만, “Quality Measurement of Object-Oriented Software". 제 1 회 소프트웨어 품질관리 심포지엄 논문집, pp115-129, 1997

[6] Albrecht, A.J.,"Measuring application development productivity. In GUIDE/SHARE: Procedings of the IBM Applications Development Symposium", Monterey, CA., pp 83-92, 1979

[7] A. J. Albrecht And J. E. Gaffney, "Software Function, Source Line of Code, and Development Effort Prediction: A Software Science Validation ", IEEE TSE, Vol 9, No.6, pp639-648, Nov 1983

[8] Bahram Adrangi and Warren Harrison, “Effort Estimation in a System Development Project", Journal of Systems Management,

pp21-23, Aug 1987
[9] Banker, R.D., Datar, S.M. and Kemerer, C.F., "A model to evaluate variables impacting productivity on software maintenance projects", Manage. Sci., 37(1), pp1-18, 1991
[10] Behrens, C.A. "Measuring the productivity of computer systems development activities with function points", IEEE TSE, 9(6), pp648-652, 1983
[11] Boehm, B. W., "Soft Engineering Economics", Prentice-Hall, 1981
[12] Dana T. Edberg And Brent J. Bowman, "User-Developed Application: An Empirical Study of Application Quality and Developer Productivity", Journal of MIS, Vol 13, No.1, pp167-185, Summer 1996
[13] Fried, Louis, "Team Size and Productivity in Systems Development: Bigger Does Not Always Mean Better", Information Systems Management, Boston, Summer 1991
[14] IFPUG,"Function Point Counting Prctices Manual Release 4.1", IFPUG, Jan 1999
[15] ISBSG,"THE BENCHMARK Release-6",ISBSG, Dec 1999
[16] Jeffery, D.R., "A Software Development Productivity Model for MIS Environments ", The Journal of Systems and Software, New York, Jun 1987
[17] Jones and Capers, "How Not to Measure Programming Quality", Computer World, Jan 20, 1986
[18] Jones, C., "A Short History of Function Points and Feature Points", Software Productivity Research Inc., Burlington, MA, 1986
[19] Jones, C., "Programming Productivity", New York, McGraw-Hill, 1986
[20] Jones, C., "Applied Software Measurement: Assuring Productivity and Quality", New York: McGraw-Hill, 1991
[21] Kemerer, C.F.,"Reliability of function points measurement: A field experiment", Communications of ACM, 36(2), pp85-97, 1993
[22] Low, G.C., and Jeffery, D.R.,"Function Points in the estimation and evaluation of the software process", IEEE TSE, 16(1), pp81-84, 1990
[23] M. Adam Mahmood, Karen J. Pettingell and Alexander I. Shaskevich, "Measuring Productivity of Software Projects: A Data Envelopment Analysis Approach", Decision Sciences, 27(1), pp57-74, Winter 1996
[24] Perry, W.E., "The best measures for measuring data processing quality and productivity", Tech. Rep., Quality Assurance Institute, 1986
[25] Rudolph, E.E., "Productivity in computer application development", Working Paper 9, Dept. of Management Studies, Univ of Auckland, Auckland, New Zealand, 1983
[26] S. Sawyer and P. J. Guinan, "Software Development: Processes and Performance", IBM Systems Journal, Vol 37, No. 4, 1998
[27] Scudder, R.A., and Kucic, R.A.,"Productivity measures for information systems", Information and Management, 20, 5, pp343-354, May 1991
[28] Tarek K., Kishore Sengupta, and Clint Swett, "The Impact of Goals on Software Project Management: An Experimental Investigation", MIS Quarterly Vol. 23 No. 4, pp. 531-555, December 1999
[29] Wilder, Clinton, "CIOs Say Downsizing Is in the Cards", Computer World, May 13, 1991