컴포넌트 기반의 개발 노력도 산정 모델

김승렬* · 서정석**

요 약

컴포넌트 개발에 보편화 추세에 있어서 이에 따른 노력도 산정 모델의 개발이 시급하지만 아직까지 간편하게 사용할 수 있는 모델이 없는 것이 현실이다. 따라서 본 연구는 그에 대한 기반을 마련하기 위하여 컴포넌트 기반에 관련된 개발 노력도 산정 모델을 살펴보고자 한다. 기존의 개발의 노력도 산정 모델로 주로 사용하고 있는 모델로는 COCOMO중심의 모델, 기능점수 중심의 모델, 그리고 상용 컴포넌트(COTS) 중심의 모델로 나누어 볼 수 있으며, 각각의 중심 모델을 비교하여 컴포넌트기반(CBD)에서 활용 가능한 모델들을 살펴보기로 한다.

1. 코코모(COCOMO) 중심의 모델

코코모 중심의 모델은 코코모 I, 코코모II, 코코모II의 재사용, 코캇스(COCOTS) 모델 등이 있으며, 각각의 모델을 살펴보면 다음과 같다.

1.1 코코모 I (Original COCOMO) 모델

오리지날 코코모(COCOMO) 불리우는 코코모I [9]은 Boehm에 의해 제안된 구조적 방법론에 입각한 소프트웨어 노력도 산정 모델로 프로젝트의 유형에 따라 크게 기본유형(Organic Mode), 반분리유형(Semi-detached Mode), 삽입유형(Embedded Mode)으로 구분할 수 있다. 기본유형은 소프트웨어 규모가 작고 프로그램 난이도가 낮으며 프로그래머 구성원의 자질이 우수한 소수 인원으로 구성된 경우이고, 반분리유형은 소프트웨어의 중요도와 프로그래머 구성원의 자질이 보통인 경우이고, 마지막으로 삽입유형은 프로그램의 난이도가 높고 소프트웨어가 제대로 작성되지 않으면 인명과 재산에 막대한 영향을 초래하여 대단히 신중을 요구하는 유형이다. 또한 코코모(COCOMO) 모델은 정확도에 따라 약식으로 노력도 산정을 할 수 있는 기본 코코모 모델(Basic COCOMO Model), 보다 상세히 노력도를 산정할 수 있는 중간 코코모 모델(Intermediate COCOMO Model), 아주 상세히 노력도를 산정할 수 있는 상세 코코모 모델(Detailed COCOMO Model) 등의 3가지 모델로 구분되어진다.

1) 기본 코코모 모델(Basic COCOMO Model)

기본 코코모 모델은 일반적으로 자체개발 등 보다 익숙한 개발 환경 하에서 중, 소규모의 소프트웨어를 개발하는데 사용되는 모델로서 소프트웨어 제품 특성, 프로젝트 특성, 컴퓨터 환경, 개발 인력의 특성 등의 속성이 고려되지 않으므로 대략적인 소프트웨어 개발 산정 및 규모를 파악하기 위하여 사용되어 진다. 그 원리는 아래 표에서와 같이 KDSI 값만 알면 <표 1-1>과 같이 COCOMO의 3가지 유형에 대하여 개발노력 및 개발기간을 예측할 수 있다. 또한 프로젝트의 규모에 따라 모델 <식 1-1>과 <식 1-2>와 같은 형식으로 표현된다.

* 국민대학교 정보관리학부 교수
** 나사렛대학교 전산정보학과 조교수

소규모 프로젝트 :

EFFORT = a * SIZE + b <식 1-1>

대규모 프로젝트 :

EFFORT = a * SIZE^b <식 1-2>

SIZE : 코드 라인 수 (단위: KDSI)

a, b : 상수

<표 1-1> COCOMO I 의 개발노력과 개발기간

개발 유형	개발 노력	개발 기간
기본유형 (Organic Mode)	PM = 2.4 * KDSI$^{1.05}$	TDEV = 2.5 * PM$^{0.38}$
반분리유형 (Semi-detached Mode)	PM = 3.0 * KDSI$^{1.12}$	TDEV = 2.5 * PM$^{0.35}$
삽입유형 (Embedded Mode)	PM = 3.6 * KDSI$^{1.20}$	TDEV = 2.5 * PM $^{0.32}$

2) 중간 코코모 모델(Intermediate COCOMO Model)

이 모델은 프로젝트의 내용이 좀더 명확해지는 개발 시점에서 모델 <식 1-3>과 같이 노력도에 대한 비용요소(Cost Driver)라고 하는 15개의 추정변수(EAF: Effort Adjustment Factor)를 BASIC COCOMO MODEL에 첨가하여 좀더 정확한 개발비용(노력도)를 예측하도록 한 모델이다. 이 모델은 투입공수의 추정치와 실제치가 비교적 정확한 모델이며, 비용특성의 수준을 효율적으로 조정함으로써 소프트웨어 개발비 추정이 <표 1-2>처럼 가능하도록 유연성을 가진 것이 특징이다.

노력도(EFFORT) = EAF * a * SIZE ^{b} <식 1-3>

<표 1-2> Intermediate COCOMO의 개발노력과 개발기간

개발 유형	개발 노력	개발 기간
기본유형 (Organic Mode)	PM = 3.2 * KDSI$^{1.05}$	TDEV = 2.5 * PM$^{0.38}$
반분리유형 (Semi-detached Mode),	PM = 3.0 * KDSI$^{1.12}$	TDEV = 2.5 * PM$^{0.35}$
삽입유형 (Embedded Mode)	PM = 2.8 * KDSI$^{1.20}$	TDEV = 2.5 * PM$^{0.32}$

3) 상세 코코모 모델(Detailed COCOMO Model)

중간 코코모(Intermediate COCOMO) 모델을 보완하여 좀더 정교한 개발 비용(노력도)을 예측하고자 할 때 사용한다. 이 모델은 미시적으로 개발의 노력도를 추정하는 형태이므로 개발 단계의 후반부에나 적용이 가능하며 모든 개발 환경이 사전에 구체적으로 정의되어야 한다. 기본 산정공식은 Intermediate 모델과 동일하나, 노력승수(EM)는 개발 공정별 노력승수에 개발 공정별 가중치를 곱하여 계산한다.

1.2 코코모II (COCOMO II) 모델

앞에서 논의한 오리지날 코코모 모델은 과거 구조적 방법론에 적용이 가능하였으나 방법론이 구조적 개발 방법론에서 객체지향 기반의 개발 방법론으로 변화함에 따라 개발 노력도 산정 모델도 오리지날 코코모에서 코코모II라는 새로운 모델로 발전된다. 코코모II는 1981년 Barry Boehm에 의하여 창시된 경제적 소프트웨어 공학(Software Engineering Economics)에서 출간된 코코모(COCOMO : Constructive Cost MOdel)을 갱신한 것으로 고속개발모델(rapid-development process models), 상용 컴포넌트 제품의 재사용 접근법, 리엔지니어링, 응용 통합(applications composition), 그리고 자동 응용 프로그램 제작 기능의 환경에서 소프트웨어 개

발 노력도를 측정할 수 있도록 만들어졌다. 또한 코코모II는 분산 미들웨어에서 지원하는 객체지향 접근법과 소프트웨어 프로세스 성숙도의 효과와 그리고 프로세스 기반의 품질 평가 등에 활용할 수 있다.

코코모II는 크게 코코모II 응용 조합 모델(Application Composition Model), 초기 디자인 모델(Early Design Model), 그리고 후기 구조 모델(Post-Architecture Model)로 분류되어지며, 각각의 특징은 다음과 같다.

1) 코코모II 응용 조합 모델
(Application Composition Model)

이 모델은 소프트웨어 생명주기 나선형(spiral) 모델의 초기 단계에서 시제품 개발시 적용할 수 있는 모델이므로 대략적인 개발 노력도 예측에 적용할 수 있다.

2) 초기 디자인 모델(Early Design Model)

이 모델은 모든 것이 명확하지 않고 소프트웨어 개발에 대한 데이터가 충분하지 않은 소프트웨어의 초기 개발 단계에서 대략적인 개발 노력도를 예측할 때 사용한다. 또한 코코모II는 기능점수를 활용하고 있으며 노력승수(EM)에서는 7가지의 비용요소를 척도요인(SF)에서는 5가지 비용요소를 제시하고 있다.

3) 후기 구조 모델(Post-Architecture Model)

이 모델은 초기 디자인 모델보다 좀더 정교한 실제적인 소프트웨어 개발 노력도와 나아가서는 유지보수 노력도까지를 측정 할 수 있는 모델이다. 이 모델은 이미 소프트웨어 개발 생명주기의 구조가 구축되었을 경우, 개발 비용 측면에서 매우 효과적이다. 규모(SIZE)를 측정하는데 있어서 재사용에 대한 변경 그리고 소프트웨어 파손에 소스 인스트럭션과 기능점수를 활용하여 측정하고 있다. 노력승수(EM)의 17가지의 비용요소와 프로젝트의 척도요인(SF)을 결정하는 5가지 비용요소가 있다.

이 세가지 모델은 현재의 개발 단계가 어디에 있느냐에 따라서 구분 적용하므로 코코모II 모델을 효율적으로 활용할 수 있는 방법의 하나이다. 코코모II에서 개발의 노력도는 PM(Man-Month)으로 표현되어지며 수식은 <식 1-4>와 같이 기술하고 있다.

$$PM_{nominal} = A * SIZE^{B} * \prod EM \quad \text{<식 1-4>}$$

여기서 A는 상수이며 규모(SIZE)는 천 단위의 코드라인수(KSLOC)을 말한다. SIZE는 조정전의 기능점수(Unadjusted Function Points : UFP)의 값에서 KSLOC의 수치로 변환이 가능하도록 설계되어졌으며, B의 값은 <식 1-5>와 같이 표현되어 진다.

$$B = 1.01 + 0.01 * \sum SF \quad \text{<식 1-5>}$$

여기서 초기 디자인(Early Design) 모델과 후기 구조(Post-Architecture) 모델의 척도요인(SF)은 경험도(Precedentedness: PREC), 개발의 유연성 (Development Flexibility: FLEX), 아키텍처/위험의 해결성 (Architecture / Risk Resolution: RELS), 팀의 결속력 (Team Cohesion : TEAM), 프로세스 성숙도 (Process Maturity : PMAT) 등 5가지 요소를 제시하고 있다.

1.3 코코모(COCOMO)II 재사용 모델

비슷한 프로젝트의 개발시 대부분의 개발 업체에서는 과거의 프로젝트를 재사용하여 개선해 가는 경우가 사실상 많이 발생한다. 이러한 부분을 고려하여 개발 노력도를 산정하고자 생긴 것이 코코모II의 재사용 모델이다. 이 모델 역시 코코모II를 기본 모델로 하고 있으며 상세한 것은 <식 1-6>, <식 1-7>, 그리고 <식 3-8>과 같다.

$$PM_{nominal} = A * SIZE^{B} * \prod EM \quad \text{<식 1-6>}$$

$$B = b1 + b2 * \sum SF \quad \text{<식 1-7>}$$

단, b1 = 1.01, b2 = 0.01

산정규모(SIZE) = <식 1-8>

$$\text{실제규모} * \frac{0.4*DM + 0.3*CM + 0.3*IM}{100}$$

여기서 DM은 디자인 변형율, CM은 코드 변형율, 그리고IM은 통합율을 나타낸다. 그러나 이러한 과정은 매우 복잡하여 실무적 차원에서 적용하기는 어려움이 많다.

1.4 코캇스(COCOTS) 모델

상용 컴포넌트 통합 노력도에 관한 모델로 코코모(COCOMO) II 모델에서 파생된 코캇스(COCOTS) 모델이 최근에 출현하였는데, 현재 코캇스 모델에서는 개발과정의 노력도 만이 측정 가능하다. 그러나 향후 코캇스 모델에서는 유지보수에 대한 노력도도 추가적으로 산출이 가능할 것으로 본다[9].

상용 컴포넌트(COTS) 제품은 몇 가지 특징을 가지고 있는데 첫째로 상용 컴포넌트를 구입한 사람은 본래의 상용 컴포넌트의 기능이나 수행 방법을 바꿀 수 없다는 점이고, 둘째로 대부분의 상용 컴포넌트 제품들은 서로 쉽게 연결할 수 없다는 점이고, 셋째로 구매자는 상용 컴포넌트의 업그레이드에 직접적으로 관여 할 수 없다는 점이며, 마지막으로 상용 컴포넌트 제작사의 특성은 다양하다는 점이다.

상용 컴포넌트 제품이 가진 위험성에는 대체적으로 상용 컴포넌트 제작사의 미성숙성, 상용 컴포넌트 사용자의 경험 미숙, 기반환경, 제품간의 비 호환성, 사용자가 상용 컴포넌트 제품에 대하여 현재 혹은 향후 변경을 할 수 없음, 대부분의 사용자가 상용 컴포넌트 제품 조립시 Plug-and-Play로 단순히 생각함 등을 예로 들을 수 있다. 그러나 이러한 위험성들은 자격 테스팅(qualification testing), 벤치마킹(Benchmarking),참조 체크(Reference checking), 호환성 분석(Compatibility Analysis) 등의 완화 단계를 통하여 줄일 수 있다.

코캇스(COCOTS) 모델은 다음과 같이 4가지의 과정을 가지고 있다.[10]

1) 상용 컴포넌트 후보 평가 및 선정 과정
2) 상용 컴포넌트 맞춤 및 투닝 과정
3) 선정된 상용 컴포넌트들의 통합 및 조립 과정
4) 조립된 소프트웨어 테스트 과정
5) 운영과 유지보수 과정

첫번째 과정으로 상용 컴포넌트 제품 평가(COTS Product Assessment) 및 선정 과정에서는 소프트웨어를 개발하기 위하여 추천된 상용 컴포넌트 제품 후보를 2단계로 선정하는 과정으로 분류되어 진다. 첫번째 단계는 초기 선정시 필요한 노력을 <식 1-9>에 의하여 계산하고, 다음 단계는 <식 1-10>과 같이 마지막 선정과정의 노력 계산한다.

초기선정노력 = 초기상용컴포넌트후보수 * 한 후보당선정노력의평균 <식 1-9>

마지막선정노력 = <식 1-10>

$$\sum_{i}^{17} \text{마지막COTS후보수} * \text{평균평가노력}$$

(단, 여기서 i는 17개의 평가 속성을 의미함)

따라서 총 상용 컴포넌트에 대한 선정 노력은 <식 1-11>과 같이 초기선정노력에 마지막 선정노력을 더함으로 얻을 수 있다.

총 상용 컴포넌트에 대한 선정노력
= 초기선정노력 + 마지막선정노력
<식 1-11>

예를 들어 상용 컴포넌트 제품 선정시 초기에 180개의 제품이 있었다고 하자. 그리고 마지막으로 상용 컴포넌트 제품을 선정할 때 그 중 20%를 선택하려고 한다면, 여기서 초기 선정노력을 계산할 때 후보 상용 컴포넌트 수는 180이고, 마지막 선정노력의 후보 상용 컴포넌트 수는 36 이 된다.

두번째 과정으로 상용 컴포넌트 맞춤 및 튜닝 과정에서는 소프트웨어 제작에 사용을 목적으로 선정된 상용 컴포넌트들을 사용 준비하는 단계로써 이 과정에 소요되는 노력도는 복잡도를 고려하여 그룹별로 맞춤 및 튜닝에 소요된 노력도에 컴포넌트 수를 곱하여 산정한다.

세번째 과정으로 상용 컴포넌트 통합과정에서는 앞에서 준비된 상용 컴포넌트들을 통합하기 위하여 사용된 접착코드수(Glue Code)를 기반으로 환경 특성을 고려하여 모델 <식 1-12>을 사용하여 노력도(PM)를 계산한다.

$$\text{노력도(PM)} = A * SIZE^{B} * \sum_{i=1}^{13} EMi \qquad \text{<식 1-12>}$$

$$B = b1 + b2 * \sum_{j=1}^{5} SFj$$

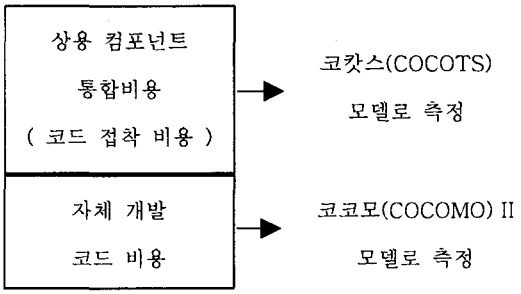
$$b1 = 1.01, \quad b2 = 0.01$$

여기서 노력승수(EM)는 13개의 노력도에 대한 비용요인은 <표 1-3>과 같다.

<표 1-3 > 노력승수(EM)의 비용요인

• 컴포넌트 제품의 성숙도
• 컴포넌트 제품에 대한 접착자의 경험
• 컴포넌트 접착자의 능력
• 컴포넌트 접착자의 통합에 대한 경험
• 컴포넌트 접착자의 지속성
• 컴포넌트 공급자의 제품 확장에 대한 의지
• 컴포넌트 제품의 상호 호환의 복잡성
• 컴포넌트 제작사의 지원
• 컴포넌트 제작사의 교육과 설명서 제공
• 개발할 응용 시스템에 대한 제한 및 하위 시스템의 신뢰성
• 개발할 응용 시스템의 호환상의 복잡성
• 컴포넌트의 기술적 성능에 대한 제약성
• 개발할 응용 시스템의 이동성

기존의 프로젝트 개발은 거의가 자체 개발 내지는 전문 업체에 위탁하는 주문 제작형이었다. 주문제작을 할 경우, 코드가 공개되어 재사용시 변형이 가능하다는 장점이 있다. 그러나 현재 상용 컴포넌트(COTS) 환경에서의 개발은 소프트웨어의 모든 부분을 부품화 개념으로 필요한 부분을 즉시 구입하여 사용하므로 가격과 시간이 많이 절약되는 장점이 있다. 하지만 상용 컴포넌트 환경이라고 모든 부품이 시장에 있는 것은 아니다. 때에 따라서는 불가피하게 자체에서 기존과 같이 개발해야만 하는 부분이 있다. 그러므로 결국 완전한 상용 컴포넌트 환경은 아직까지 현실에 맞지 않으므로 개발 노력도를 예측할 때는 <그림 1-2>와 같이 두 가지 모형으로 분리하여 측정하여야 한다. 그러나 이것은 과다한 정보를 요구하므로 업계의 현실에 맞지 않는다는 점이 이 모델의 문제점으로 지적되고 있다.



<그림 1-2> 분류된 개발 노력도 측정시 모델 적용

이상의 코코모 중심의 모델을 요약 하면 <표 1-4>와 같다.

<표 1-4> 코코모 중심 모델의 비교

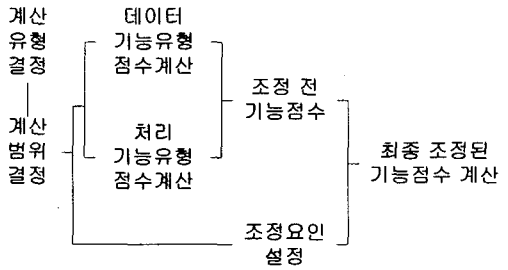
모델	모형
코코모I	$EAF * a * SIZE^b$
코코모II	$PM = A * SIZE^B * \sum_{i=1}^{17} EMi$ $B = 1.01 + \sum_{j=1}^{5} SFj$
코코모 II 재사용 모델	$PM = A * ESIZE^B * \sum_{i=1}^{17} EMi$ $ESIZE = SIZE * [(0.4DM + 0.3CM + 0.3IM)/100]$
코캇스 (COCOTS)	$PM = A * SIZE^B * \sum_{i=1}^{13} EMi$

코코모 중심의 모델에 대한 평가는, 첫번째로 프로젝트 개발 초기 상태에서는 규모(SIZE)의 예측이 어렵다는 점이다. 코코모 모델의 가장 근본이 되는 것은 규모(SIZE)인데, 이 규모는 프로그램 소스코드의 라인 수를 기준으로 하고 있다는 점이다. 두번째로 성공적인 예측을 하기 위해서는 모델에서 요구하는 과거의 데이터들이 있어야 하는데 현실은 그러하지 못하다는 점이다[11].

2. 기능점수(FP) 중심의 모델

2.1 기능점수(FP) 모델

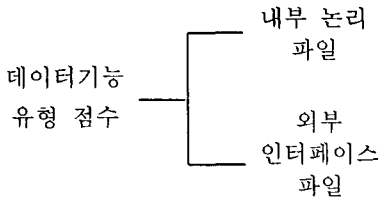
기능점수는 현재 1994년 재정된 표준을 사용하고 있으며, Albrecht모형을 근간으로 이루어져 있다. 이 중 기능점수 분석(FPA : Function Point Analysis)은 소프트웨어 제품의 기능적 크기를 측정(measurement)하기 위해 설계된 기술이며, 사용자의 관점에서 소프트웨어 제품을 측정하고자 하는 방법으로, 현재 MIS영역에서 널리 사용되며, 사실상의 표준(de facto)으로 자리잡고 있다. 최근 기능점수 분석(FPA)의 버전 4.0은 ISO의 소프트웨어공학(Software Engineering)분과에 소속되어 국제표준으로써 자리를 잡고 있다. 하지만, 기능점수 분석은 MIS 영역 이외의 영역 즉, real-time Software에서는 잘 적용되고 있지 못하는 단점을 지니고 있다. 이를 개선하기 위해, IFPUG내에서는 FFP(Full Function Point)를 계발 연구 중에 있으며, 많은 부분에서 기존 기능점수를 보완하고 있다.



<그림 2-1> 기능점수 계산 절차

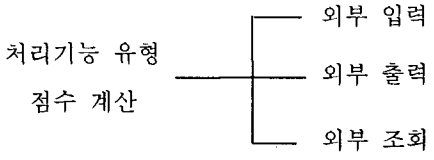
기능점수를 계산하는 계산 절차는 <그림 2-1>과 같이 가장 먼저 계산유형을 결정하고 그 유형에 대한 범위를 결정하고 그 범위 안에서의 데이터 기능유형과 처리 기능유형의 점수를 계산하면 조정전의 기능점수가 나오는데, 이 값에 기술적 복잡도를 감안한 조정요인을 고려하여 최종 조정된 기능점수 값을 계산한다.

세부적으로 가장 먼저 해야 할 계산 유형의 결정은 개발 프로젝트 기능점수, 확장 프로젝트 기능점수, 응용 프로젝트 기능점수 등의 3가지 유형 중 해당하는 유형을 선택한다. 이 선택된 유형의 계산 범위를 결정하고 데이터 기능 유형에 대한 점수를 <그림 2-2>와 같이 내부논리파일(LIF)과 외부인터페이스파일(EIF)로 나누어 정해진 테이블을 기준으로 계산한다.



<그림 2-2> 데이터 기능 유형 계산

다음으로 처리기능 유형을 계산하기 위하여 <그림 2-3>와 같이 외부입력, 외부출력, 외부조회 등 3가지로 나누어 자료, 제어정보, FTR, DET 등의 기본 자료 각각에 대한 복잡도와 기여도 점수를 기준 테이블에 의하여 계산 한다.



<그림 2-3> 처리기능 유형

이상에서 구한 데이터 기능유형 점수와 처리 기능유형 점수 합면 조정 전의 기능점수 값이 나오는데, 이 값에 조정요인의14가지 기술적 복잡도를 감안하면 다음과 같은 식에 의하여 최종 조정된 기능 점수 값을 구해진다.

$FPA_{Aadjusted} = 0.65 + 0.01 * TDI$ <식 3-13>

$FPA_{Aadjusted}$: 조정된 기능점수
TDI : 기술적 요소(14개)

이상의 결과로 처음에 정의한 개발 프로젝트 기능점수, 확장 프로젝트 기능점수, 응용 프로젝트 기능점수 등의 3가지 유형에 따라 <표 2-1>와 같은 유형을 적용하여 계산 한다[12].

<표 2-1> 기능점수에 의한 개발 프로젝트 유형

프로젝트 유형	모델 식
개발프로젝트 (Development Project)	DFP = (UFP + CFP) * VAF
	UFP : 조정전 기능점수 CFP : 처리유형의 기능 점수 VAF : 기술적 복잡 요소의 의한 값
확장 프로젝트 (Enhancement Project)	EFP =[(ADD+ CHGA+ CFP)*VAFA]+ (DEL*VAFB)
	ADD :확장 프로젝트 보정전 기능점수 CHGA : 확장 프로젝트 변경후 보정전 기능점수 CFPB : 확장 프로젝트에서 변경전의 보정전 기능점수 DEL : 확장 프로젝트에서 삭제된 보정전 기능점수 VAFB : 기술적 복잡도의 보정전 기능점수 VAFA : 기술적 복잡도의 보정후 기능점수
응용 프로젝트 (Application Project)	변경 요구 없을 때 AFP = ADD * VAF 변경 요구 있을 때 AFP = [(UFPB+ ADD+ CHGA) - (CHGB+ DEL)]*VAFA
	UFPB : 확장프로젝트 시작 전의 보정전 기능점수 CHGA : 확장 프로젝트에서 변경후의 보정전 기능점수 CHGB : 확장 프로젝트에서 변경전의 보정전 기능점수 VAFA : 기술적 복잡도의 보정후 기능점수

이러한 기능점수는, 예를 들면 리얼타임과 같은 경우, 즉 많은 양의 자료가 동시 다발적으로 값이 변할 때 ILFS와 EIFS를 통해서 그룹 화하는 것은 어렵다는 단점과 프로그램이 다수의 서브 프로세스(Sub-Process)로 이루어 졌을 때 측정하기 곤란하다는 단점을 지니고 있다. 이러한 기능점수의 문제점을 해결하기 위해 최근 기능점수의 확장 모델인 완전 기증점수(FFP) 모델이 제안되고 있다.

2.2 완전 기능점수(FFP) 모델

FFP(Full Function Point)는 기존의 기능점수 분석(FPA)을 관리 기능점수(Management FP)로 국한하고, 새로운 기능점수인 제어기능점수

(Control FP)를 결합한 형태로 관리형태의 데이터(프로세스)인지, 제어형태의 데이터(프로세스) 인지 데이터(프로세스) 그룹을 식별하여야 한다. 여기서 식별된 데이터와 프로세스는 데이터와 처리 기능 유형에 따라 나누게 되는데 이중 관리적 데이터와 처리는 기능점수 분석(FPA)방법을 따르며, 제어 데이터와 제어 처리는 별도의 규칙에 따라 기능점수를 계산한다. 여기서 관리적 데이터와 처리를 관리적 기능점수(Management FP)라고 하며, 제어데이터와 제어처리를 제어 기능점수라고 한다. 특히, 관리적 기능점수는 기존 기능점수에서 제어정보를 삭제한 형태이다. 이상의 정리하면 <식 2-2>와 같이 요약된다[1][2][4].

FFP = MFP + CFP = (FPA- CI) + CFP
<식 2-2>

FFP : 완전기능점수
MFP : 관리기능점수
CFP : 컨트롤기능점수
FPA : 기능점수 분석
CI : 컨트롤 정보

현재 FFP모형은 초기 정착단계에 있으며, 향후 IFPUG에서 표준으로 채택하여 사용할 가능성이 가장 높은 모형으로 실제 모형개발에 참여하였던 연구자들은(Jean-Marc Desharnais, Denis St-Pierre, Marcela Maya, Alain Abran) 국제 학술회의에서 모형의 당위성을 증명하고 있으며, 실제 Filed Test에서도 만족할 만한 결과를 도출한 것으로 발표되고 있다. FFP모형의 경우, 기존의 경영정보 영역의 소프트웨어에 대한 기능점수 분석(FPA)보다는 약간 높은 기능점수(약 2%~5%)를 나타내고 있으나, 기존의 FPA에서는 계산하기 힘든, 업무분야 외의 소프트웨어에 비교적 잘 적용되고 있는 것으로 나타났다[5][7].

이상의 기능점수 중심의 모델은 <표 2-2>과 같이 요약 할 수 있다.

<표 2-2> 기능점수 중심의 모델

모델	관계 식
기능점수	$FPA = \sum_{i=1} (FCi * Wi)$ $FPA_{adjusted}$ = FPA * (0.65 + 0.01 * TDI)
완전기능점수	FFP = MFP + CFP = (FPA - CI) + CFP

3. 상용 컴포넌트(COTS) 중심의 모델

3.1 R. Stutzke 모델

R. Stutzke는 COTS를 활용한 소프트웨어의 개발 비용의 추가비용으로 <식 3-1>와 같은 모델을 제시하였다[6].

비응집력에 의한 추가비용 =
CV * AC * IS * (CS + CC) <식 3-1>

CV(Component Volatility) :
상용 컴포넌트의 새로이 개발된 버전 수
AC(Architectural Coupling) :
개발할 소프트웨어에 사용될
상용 컴포넌트 수
IS(Interface Size) :
상용 컴포넌트와 연결할 프로시져, 함수,
엔트리포인트 등의 수
(단, Passing Parameter의 수에
가중치를 부여함)
CS(Cost of Screening) :
사용할 상용 컴포넌트를 선별하는 비용
CC(Cost of making Changes) :
컴포넌트들의 사용 변경 비용

이 모델은 소프트웨어를 개발할 때 상용 컴포넌트를 사용한 개발 비용면을 보기보다는 상용 컴포넌트의 새로운 버전으로 변경됨으로 인한 소프트웨어 개발의 추가 비용면을 보여주고 있다. 또한 실제로 이 모델에 맞추어 데이터로 구현하여 추가 비용을 산정하지는 못하였다.

3.2 M. Karpowich 모델

M. Karpowich는 상용 컴포넌트를 활용한 소프트웨어의 개발 비용으로 <식 3-2>과 같은 식을 제시하였다[3].

상용 컴포넌트 통합비용 =
(상용 컴포넌트 라이세스비 * 라이센스수)
+ (상용 컴포넌트 제품 사용 교육비)
+ (상용 컴포넌트 접착비) <식 3-2>

이 모델은 소프트웨어를 개발할 때, 상용 컴포넌트를 3가지 측면으로 분류하여 비용을 산정하였다. 하나는 상용 컴포넌트를 사용할 때 발생하는 사용료와 둘째로 상용 컴포넌트 제품을 사용할 수 있도록 하는 교육에 대한 비용, 그리고 마지막으로 개발하고자 하는 소프트웨어 안에 상용 컴포넌트 제품들을 서로 연결하고 통합하여 사용하기 위한 상용 컴포넌트 접착비용이다.

이 모델에서 비용 산정에 중요한 요인들을 제시하고 있지만 상용 컴포넌트 접착비에 대한 비용 측정을 상당부분 무시하고 있는 것이 문제이다.

3.3 Loral COTS 모델

T. Ellis <식 3-3,4,5>와 같은 모델을 제안하였다[8].

WU(Work Units) = fn(Size, Drivers) <식 3-3>
P(Productivity) =
Labor-Months/Work Unite <식 3-4>
Estimated Effort in LM = WU * P <식 3-5>

Size : 기능점수에 의한 COTS접착의
코드 규모
Drivers : 17가지의 드라이버의 통합 비용
LM : Laber-Months
P : 생산성
WU : 워크 유니트

이 모델은 17가지의 드라이버 통합 비용(노력도)에 중심을 두고 있으며, 워크 유니트를 규모와 드라이버의 함수로 표현한다. 이렇게 구해진 워크 유니트는 생산성이라는 단위를 곱하여 노력도를 측정한다. 이 모델은 소프트웨어 개발의 구현 단계에 잘 적용이 되며, 오차범위 15%이내의 노력에 대한 개발비용을 예측할 수 있다. 여기서 중요한 것은 상용 컴포넌트에 대한 비용요인(Driver)인데, 그 요인은 다음과 같은 것들이 있다.

- 제품 성숙성(Product Maturity)
  시장에 출품된 기간, 업그레이드 및 테스트의 유무, 제품의 시장 점유율, 오류의 교정 정도, 시장의 표준정도.
- 개발자의 성숙도(Vendor Maturity)
  개발자가 시장에 참여한 기간, 명성, 제품의 다양성
- 구성과 맞춤제작
  (Configurability/Customization)

구성 선택의 수, 제품을 맞춤제작하기 위하여 필요한 노력도

- 설치의 용이성(Installation Ease)
  재품 설치에 필요한 노력도
- 업그래이드의 편이성(Ease to Upgrade)
  컴포넌트 업그래이드 시 개발하고 있는 소프트웨어에 얼마나 쉽게 적용하여 재 조립 할 수 있나 하는 정도.
- 제작사의 협조도(Vendor Cooperation)
  컴포넌트 제작사가 사용자의 조언에 따라 컴포넌트를 개량 또는 제안을 받아들일 의지도. 제작사의 협조도가 높으면 높을수록 사용자는 새로운 개발을 줄이고 컴포넌트를 접착할 코드도 쉬워진다.
- 제품 서포트의 서비스(Product Support Service)
  제작사의 24시간 서비스, 세미나, 문제점 지적 등을 통한 서비스의 종류
- 제품 서포트의 질(Product Support Quality)
  사용자 질문에 대한 제작사의 의무감
- 사용자, 관리자, 그리고 설치에 대한 설명서 (User, Administrator, &.Installation Documentation)
  사용자, 관리자, 그리고 설치에 대한 설명서의 질
- 사용자의 편이성(Ease of Use for End User)
  사용자에게 직관적으로 얼만한 편의를 제공하는 편이성
- 관리자의 사용 편이성(Ease of Use for Administrator)
  개발 관리자에게 직관적으로 얼만한 편의를 제공하는 편이성
- 사용자와 관리자의 교육(End User & Administrative Training)
  제공하는 교육의 질과 종류
- 관리자의 노력(Administrative Effort)
  시스템의 유지보수에 관리자가 얼마만한 시간을 투자하는 노력
- 호환성(Portability)
  다른 플렛폼 상에서의 제품 호환성
- 제품에 대한 경험(Previous Product Experience)
  컴포넌트 제품에 대한 개발/사용/통합 경험성
- 제품 업그래이드 빈도(Expected Release Frequency)
  업그래이드에서 완성된 제품으로 배포하는데 걸리는 시간.
- 응용 소프트웨어 혹은 시스템 컴포넌트 패키지인가
  컴포넌트 제품이 응용형 인가 혹은 시스템형 인가?

이 모델의 문제점은 코드가 접착 코드와 일반 소스코드가 혼존 함을 무시하였고 노력도에 대한 비용요소가 17가지나 되므로 실질적으로 이러한 데이터를 예측하기가 어렵다는 점이다.

3.4 USC COTS 모델

USC(University of Southern California)에서 개발한 상용 컴포넌트 통합 노력도를 연구한 모델이 있는데 편이상 우리는 USC COTS 모델이라 부르기로 한다. 이 모델은 코코모 모델을 기반으로 Loral COTS 모델의 노력에 대한 비용요소들을 추가 삭제하여 개선한 모델이다. USC COTS모델은 코코모II의 재사용 모델을 기반으로 하고 있으며 <식 3-6>과 같이 표현된다.

$$PM = A * (ESIZE)^{B} * \prod_{i=1}^{13} EMi \qquad \text{<식 3-6>}$$

ESIZE = UFP * (1.0 + BRAK / 100)
COST = PM * ($$/PM)
UFP : 기능점수로 예측한 코드라인수
PM (노력도) : 참여 인원수 * 월
ESIZE : 예상 규모
BRAK : 코드 프렉키지 (%)
EM : 노력승수 COST : 비용(달러)

노력승수(EMi)의 비용요소는 Loral COTS모델과 COCOMO II의 환경 요인을 복합하여 <표 3-1>과 같은 13가지의 새로운 노력승수의 비용요소를 만들었다.

<표 3-1> USC COTS모델의 노력승수에 대한 비용요소

- 상용 컴포넌트 제품 및 매뉴얼의 성숙도(CPDM)
- 상용 컴포넌트 공급자의 제품 확장에 대한 의지(CVEW)
- 상용 컴포넌트 제품에 대한 접착자의 경험(CIEP)
- 상용 컴포넌트의 신뢰성(CREL)
- 상용 컴포넌트 제품과 응용에 대한 복잡성(CPAX)
- 상용 컴포넌트 접착자의 능력(CIPC)
- 상용 컴포넌트 통합 구조 및 위험부담 해결성(CIAR)
- 상용 컴포넌트의 컴포넌트의 개발 호환의 표준에 대한 순응성 (CCOS)
- 상용 컴포넌트의 성능(CPER)
- 상용 컴포넌트 통합자의 COTS통합에 대한 경험(CIXI)
- 상용 컴포넌트 제공자의 성숙도와 제품에 대한 지원(CVMS)
- 상용 컴포넌트 제공자의 교육(CVPT)
- 상용 컴포넌트 이동성(CPRT)

계산 절차는 가장 먼저 접착코드의 크기 만을 측정하고, 둘째로 코드에 대한 브랙이지(Breakage)를 측정한다. 세번째로 규모(ESIZE)를 측정하고, 네번째로 노력승수(EM)를 측정하여 노력도(PM)을 계산해 낸다. 이렇게 산출된 노력도는 상용 컴포넌트에 대한 통합 비용에 해당하므로 여기에 다시 코코모II를 사용하여 일반 코드에 대산 규모도 감안해 주어야 한다.

이 모델의 문제점은 비례요인(SF)의 B값을 1로 가정 하였으며, 현재 B에 대한 보안 작업을 하고 있다. 또한 접착(통합)코드가 깨질 확률(Brak/100)을 소프트웨어 개발 초기 단계에서 예측하기가 어렵다는 점이고, 노력승수의 비용요소가 13가지나 되므로 이러한 데이터의 예측이 어렵다는 점이다.

이상의 상용 컴포넌트(COTS) 중심의 모델을 요약하면 <표 3-2>과 같다.

<표 3-2> 상용 컴포넌트 중심의 모델

모델	관계 식
R. Stutzke 모델	비응집력에의한 추가비용 = CV * AC * IS * (CS + CC)
M. Karpowich 모델	COTS통합비용 = (COTS라이세스비*라이센스수) + (COTS제품 사용 교육비) + (COTS접착비)
Loral COTS 모델	WU(Work Units) = fn(Size, Drivers) P(Productivity) = Labor Months/Work Unite Estimated Effort in LM = WU * P
USC COTS 모델	ESIZE = UFP * (1.0 + BRAK/100) $PM = A * (ESIZE)^{B} * \sum_{i=1}^{13} EMi$ COST = PM * ($$/PM)

4. 기존 모델의 문제점

앞에서의 기존 연구 모델을 본 연구 모델에 적용시 발생되는 문제점들을 요약해 본 결과 <표 4-1>와 같다.

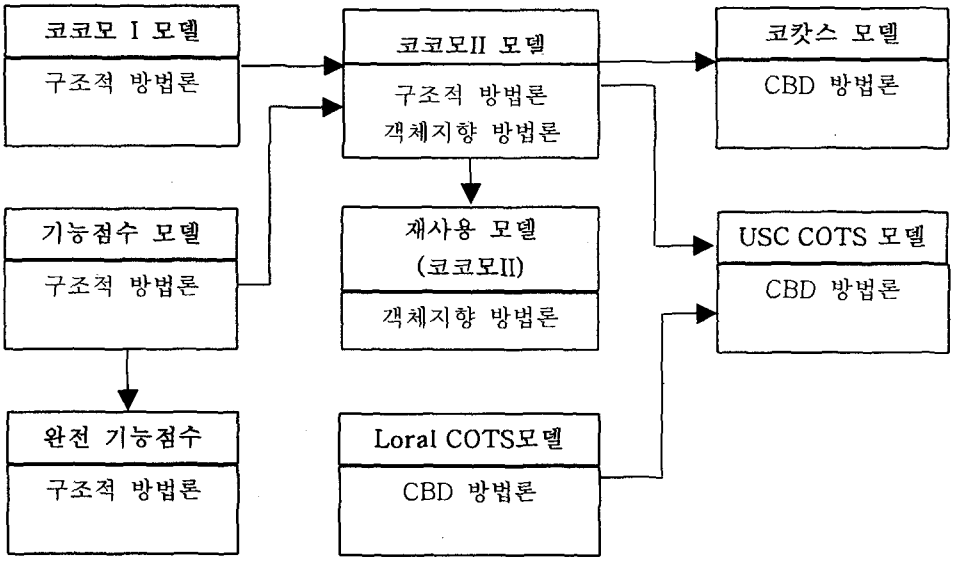
<표 4-1> 기존 연구 모델의 문제점

모델 유형	모델 종류	문제점	기반
코코모 중심 모델	코코모 I	• 구조적 방법론	구조적
	코코모II	• 코드라인 수 예측이 어려움 • 비용요인 측정이 복잡함 • 자료 수집이 어려움	객체지향
	재사용	• 코드라인 수에 대한 측정이 복잡함 • 자료 수집이 어려움	객체지향
	코캇스	• 접착코드와 소스코드 비용을 분리 측정해야 함 • 자료 수집이 어려움	CBD
기능점수 중심 모델	기능점수	• 측정이 복잡함 • 자료 수집이 어려움	객체지향
	완전기능점수	• 측정이 복잡함 • 자료 수집이 어려움	객체지향
상용 컴포넌트 중심 모델	R. Stutzke	• 비접착에서 발생하는 추가비용 만산정	CBD
	M. Karpowich	• 컴포넌트 접착비용에 대한 설명 불충분	CBD
	Loral COTS	• 측정이 복잡함 • 자료 수집이 어려움	CBD
	USC COTS	• B = 1로 가정 • 측정이 복잡함 • 자료 수집이 어려움	CBD

5. 기존 모델의 상호 관계와 방법론

구조적 방법론을 기반으로 한 코코모 I 모델은 후속 모델인 코코모 II 모델의 기반이 되며, 코코모 II 모델은 객체지향 방법론을 수용하는 모델로써 재사용 모델의 기반이 된다. 또한 규모(SIZE)의 여측이 어려우므로 기능점수 모델의 기능 점수를 규모를 측정하는 방법으로 사용하기도 한다.

코코모 II 모델은 컴포넌트기반(CBD)의 노력도를 산정하기 위하여 발전한 것이 코캇스(COCOT) 모델이다. 컴포넌트 기반의 노력도를 측정하기 위한 가장 최근의 모델로써는 Loral COTS 모델과 코코모 II 모델을 혼합한 USC COTS 모델이 있으며, 상세한 기존 모델의 상호 관계와 방법론을 요약하면 그림 <4-1>과 같다.



<그림 4-1> 기존 모델의 상호 관계와 방법론

참고 문헌

[1] D. St-Pierre, M. Maya, A. Abran, and J.M. Desharnais, "Adapting Function Points to Real-time Software", IFPUG Fall Conference, 1997

[2] D. St-Pierre, M. Maya, A. Abran, J.M. Desharnais,and P. Bourque, "Full Function Points : Counting Practices Manual", SEMRL/SELAM, CA, 1997.

[3] M. Karpowich, T. Sanders and R. Verge, "An Economic Analysis Model for Determining the Custom vs. Commercial Software Tradeoff," in T. Gulledge and W. Hutzler, Analytical Methods in Software Engineering Economics, Springer-Verlag, 1993.

[4] M. Maya, A. Abran, S. Oligny, D. St_Pierre, and J.M. Desharnais, "Measuring the funtionla size of real-time software", SELAM, 1997

[5] P. Morris and J.M Desharnais, "Measuring ALL the software not just what the business Uses", Total Metrics and SELAM, 1998

[6] R. Stutzke, "Costs Impact of COTS Volatility, "Knowledge Summary: Focused Workshop on COCOMO 2.0, USC Center for Software Engineering , May 16~18, 1995.

[7] S. Oligny, A. Abran, J. M. Desharnais, and P. Morris, "Functional size of real time software : overview of field tests", UQAM SEML, 1998.

[8] T. Ellis, "COTS Integration in Software Solutions - A Cost Model," in Systems Engineering in the Golbal Marketplace, NCOSE International Symposium, St. Louis, MO, July 24~26, 1995.

[9] http://sunset.usc.edu/COCOMOII/suite.html

[10] http://sunset.usc.edu/COCOTS/cocots.html

[11] http://www.ecfc.u-net.com/cost/cocomo.htm

[12] http://www.ifpug.org