3.2 노력 추정

- **소프트웨어 개발 비용 예측**
  - **정확한 비용 예측은 매우 어려움**
    - 알려지지 않은 요소가 산재
    - 원가의 계산이 어려움
  - **과거의 데이타가 필요**
  - **단계적 비용 산정 방법도 사용**
- **예산** – 세부예산 별기
  - **인건비:** MM(인원/월)을 기초
  - **경비:** 여비, 인쇄비, 재료비, 회의비, 공공요금
  - **간접 경비:** overhead (예, 사무실운영비 등)

비용에 영향을 주는 요소

- 제품의 크기
  - 제품의 크기가 커짐에 따라 기하급수로 늘어남
- 제품의 복잡도
  - 응용 : 개발지원 : 시스템 = 1 : 3 : 9
- 프로그래머의 자질
  - 코딩, 디버깅의 능력차
  - 프로그래밍 언어, 응용 친숙도
- 요구되는 신뢰도 수준
- 기술 수준(개발 장비, 도구, 조직능력, 관리, 방법론 숙달)

비용에 영향을 주는 요소

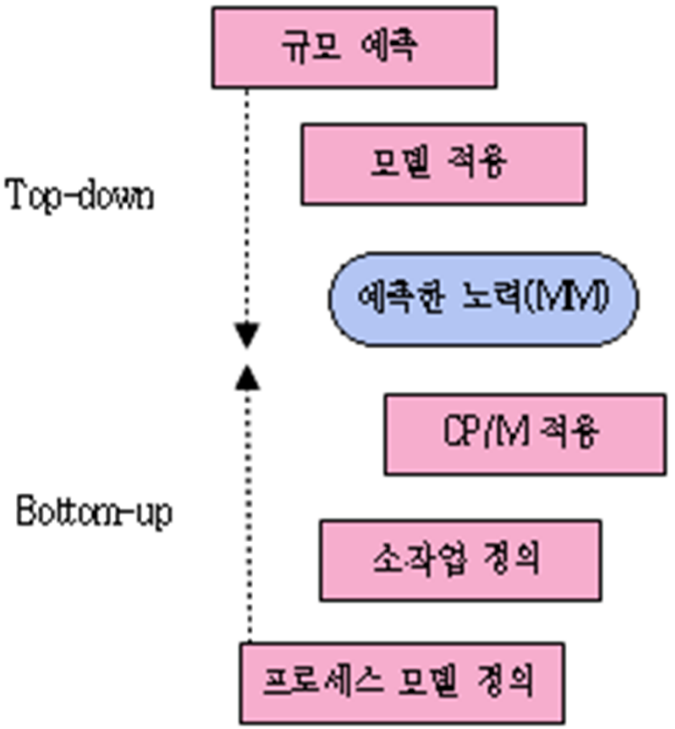
- 남은 시간
  - Putnam “프로젝트의 노력은 남은 개발 기간의 4제곱에 반비례”

-- **B** is a scaling factor and is a function of the project size.
– Productivity is the process productivity.
– Total effort decreases as the time to complete the project is extended.

$$\text{Effort} = \left[\frac{\text{Size}}{\text{Productivity} \cdot \text{Time}^{4/3}}\right]^3 \cdot B$$

프로젝트 비용을 예측하는 방법

- **상향식**
  - 소작업별 소요 기간을 구하고 여기에 투입되어야 할 인력과 투입 인력의 참여도를 곱하여, 최종 인건 비용을 계산
  - 소작업에 대한 노력을 일일이 예측
- **하향식**
  - 프로그램의 규모를 예측하고 과거 경험을 바탕으로 예측한 규모에 대한 소요 인력과 기간을 추정
  - 프로그램의 규모: LOC, **기능 점수**



COCOMO (Constructive Cost Model) 방법

- Boehm이 개발 : **노력 = A * (Size)^B * M**
  - TRW의 2K-32K 정도의 많은 프로젝트의 기록을 통계 분석
- 표준 산정 공식

	노력(MM)	기간(D)
유기형	PM = 2.4*(KDSI)**1.05	TDEV=2.5*(PM)**0.38
반결합형	PM = 3.0*(KDSI)**1.12	TDEV=2.5*(PM)**0.35
내장형	PM = 3.6*(KDSI)**1.20	TDEV=2.5*(PM)**0.32

유기형: 5만 라인 이내,
반결합형: 30만 라인 이내

•KDSI- Thousands of Delivered Source Instruction. KLOC와 같은 의미.

KSDI

- DSI(Delivered Source Instructions), which are very similar to SLOC. Cocomo II uses SLOC.
- SLOC is defined such that:

  - Only Source lines that are DELIVERED as part of the product are included -- test drivers and other support software is excluded

  - SOURCE lines are created by the project staff

    → code created by applications generators is excluded

  - One SLOC is one logical line of code

    ex) an If-Then-Else (appearing three lines) is one SLOC, but multiple DSIs.

  - Declarations are counted as SLOC

  - Comments are not counted as SLOC

* **SLOCCount : a** tool to count SLOC

```
/* Now how many lines of code is this? */
for (i = 0; i < 100; i++)
{
    printf("hello");
}
```

➔ SLOC 2줄

```
for (i = 0; i < 100; i++) printf("hello"); /* How
many lines of code is this? */
```

Physical SLOC ? ➔1

(Logical) SLOC ? ➔ 2

COCOMO (Constructive Cost Model) 방법

- 예
  - CAD 시스템　　예상 규모: 33360 LOC

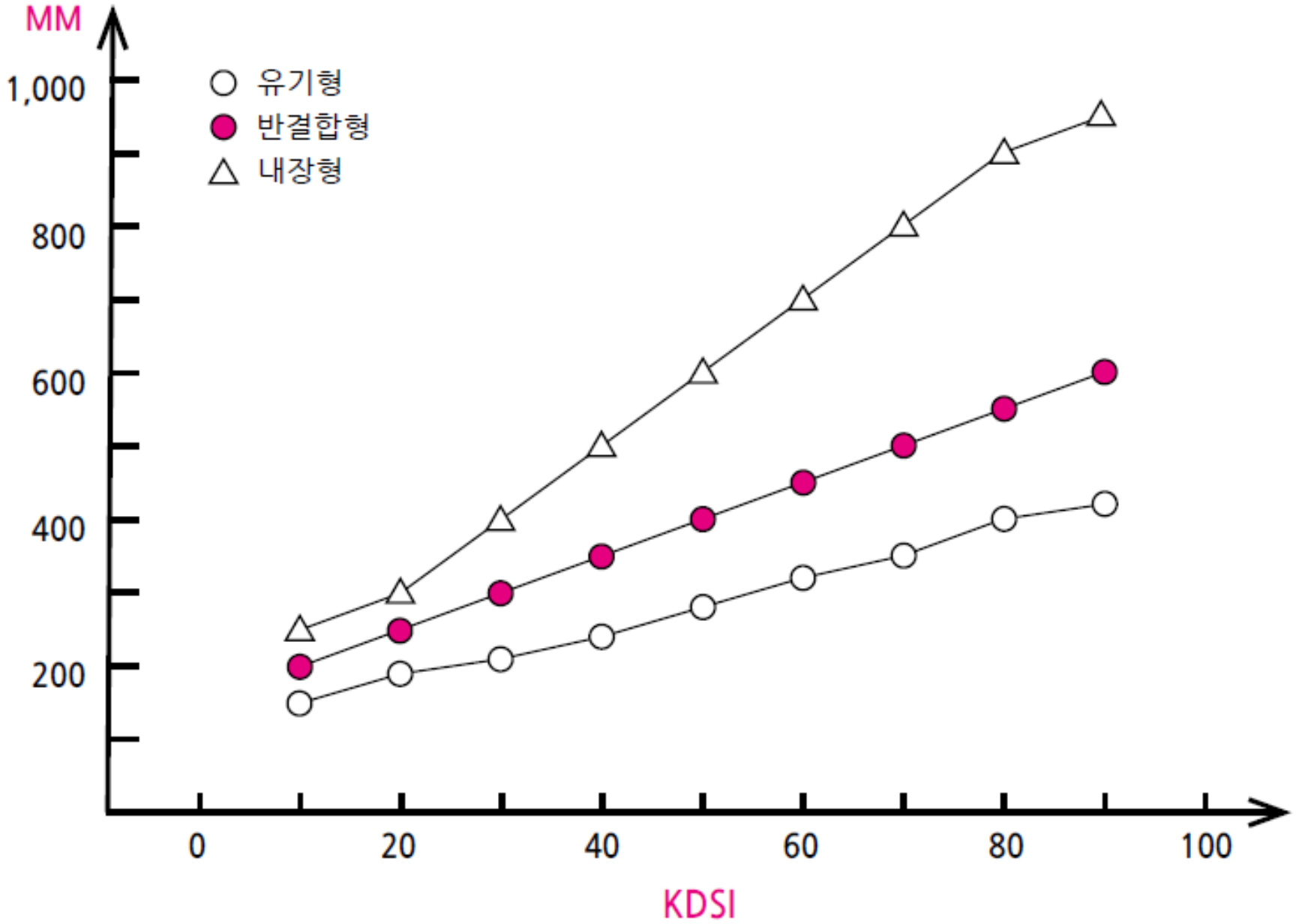
    PM = 3.0*(KDSI)**1.12 = 3.0*(33.3)**1.12 = 152MM

    TDEV = 2.5*(PM)**0.35 = 2.5*(152)**0.35 = 14.5 M

    N=E/D = 152/14.5 ~ 11 명
  - 보정

약 30K로서 반결합 형 으로 판단.

COCOMO에 의한 비용 예측



COCOMO-81 방법

모델	내용	기타
기본 COCOMO (Basic COCOMO)	추정된 LOC를 프로그램 크기의 함수로 표현해서 소프트웨어 개발 노력(그리고 비용)을 계산.	S/W 크기와 개발 모드
중간급COCOMO (Intermediate COCOMO)	프로그램 크기의 함수와 제품, 하드웨어, 인적 요소, 프로젝트 속성의 주관적인 평가를 포함하는 "비용 유도자(cost driver)"의 집합으로 개발 노력을 계산	15개의 비용 요소를 가미하여 곱한 가중치 계수 이용
고급 COCOMO (Advanced COCOMO =Detailed COCOMO)	소프트웨어공학 과정의 각 단계(분석, 설계등)에 비용 유도자의 영향에 관한 평가를 중간급 모형의 모든 특성을 통합시킨 것.	시스템을 모듈, 서브 시스템으로 세분화한 후 Intermediate와 동일

Detailed Cocomo:

- the whole software is divided into different modules and then we apply COCOMO in different modules to estimate effort and then sum the effort.

- The effort is calculated as a function of program size and a set of cost drivers are given according to each phase of the software lifecycle.

중간 COCOMO 방법

	15 비용 드라이버	비율					
		매우낮음	낮음	보통	높음	매우높음	극히매우높음
제품특성 요구	RELY	0.75	0.88	1	1.15	1.4	
	DATA		0.94	1	1.08	1.16	
	CPLX	0.7	0.85	1	1.15	1.3	1.65
H/W 제약	TIME			1	1.11	1.3	1.66
	STOR			1	1.06	1.21	1.56
	VIRT		0.87	1	1.15	1.3	
	TURN		0.87	1	1.07	1.15	
개인특성 능력/경험	ACAP	1.46	1.19	1	0.86	0.71	
	AEXP	1.29	1.13	1	0.91	0.82	
	PCAP	1.42	1.17	1	0.86	0.7	
	VEXP	1.21	1.1	1	0.9		
	LEXP	1.14	1.07	1	0.95		
PROJECT 특성	MODP	1.24	1.1	1	0.91	0.82	
	TOOL	1.24	1.1	1	0.91	0.83	
	SCED	1.23	1.08	1	1.04	1.1	

Runtime values for performance

VIRT: 가상환경의 안전성(변화빈도)
ACAP :분석능력
AEXP: 응용능력
VEXP: 컴퓨터(가상환경) 친숙성

MDOP: SE 방법 사용정도
TOO: 툴 사용정도

PCAP: the estimation of capability of the programmers are ability, efficiency, diligence, and the ability of information exchanges and collaborations.

중간 COCOMO 방법

- 모든 노력 승수를 곱한다.
  - 예: $E=EAF * 2.4(32)^{1.05} = EAF * 91$ man-months
- 단점
  - 소프트웨어 제품을 하나의 개체로 보고 승수들을 전체적으로 적용시킴
  - 실제 대부분의 대형 시스템은 서로 상이한 서브 시스템으로 구성되며 이중 일부분은 Organic Mode이고 다른 부분은 Embedded Mode인 경우도 있다.

COCOMO II

- **1995년에 발표**
- **기본 모델** $E = b * S^c * m(X)$, where $c = 1.01 + 0.01*\sum W_i$

(http://cs.uns.edu.ar/~prf/teaching/APS18/downloads/PRACTICA/COCOMO2summary.pdf)

- **소프트웨어 개발 프로젝트가 진행된 정도에 따라 세가지 다른 모델을 제시**
  - **1 단계**: 프로토타입 만드는 단계 (Prototyping, Application Composition)
    - 화면이나 출력 등 사용자 인터페이스, 3 세대 언어 (3GL) 컴포넌트 개수를 세어 응용 점수(application points)를 계산
    - 이를 바탕으로 노력을 추정
  - **2 단계**: 초기 설계 단계 (Early Design)
    - 자세한 구조와 기능을 탐구
  - **3 단계**: 구조 설계 이후 단계 (Post Architecture)
    - 시스템에 대한 자세한 이해

COCOMO II

- Application composition model. Used when software is composed from existing parts.
- Early design model. Used when requirements are available but design has not yet started.
- Reuse model. Used to compute the effort of integrating reusable components.
- Post-architecture model. Used once the system architecture has been designed and more information about the system is available.