

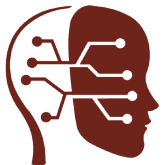
Software Engineering

Lecture 04: 계획 (프로젝트 관리와 계획) (Part 1)

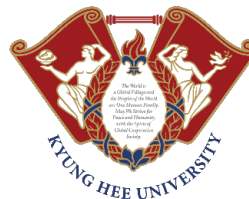
Professor: Jung Uk Kim

ju.kim@khu.ac.kr

Computer Science and Engineering, Kyung Hee University

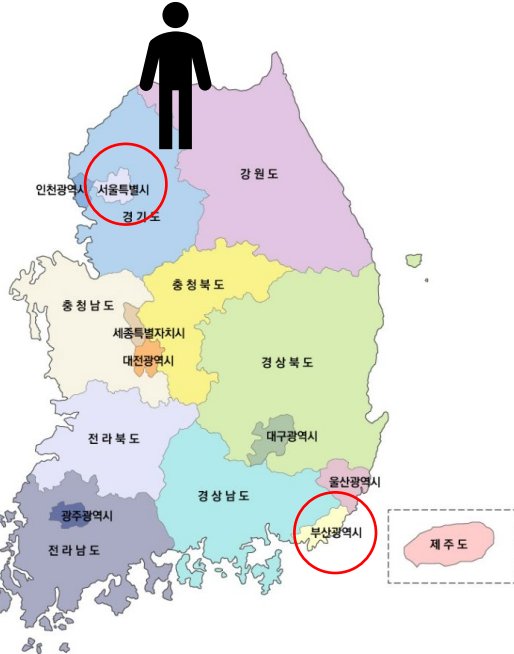


Visual AI Lab.



계획

• 부산을 가자 (출발은 서울로 가정)



부산을 가기 위한 사전 계획

(1) 어디를 가는가? → 부산

(2) 내가 부산을 가는 것이 맞나? 가진 돈은 충분한가?

(3) 숙박비용은 얼마? 교통비용은 얼마? (원화 계산, 달러 계산?)

(4) 일정은 어떻게 되는가? (3박4일, 4박5일), 아침에 출발? 저녁에 출발?

(5) 해수욕장에 해파리는 없을까? 사고가 나지는 않을까?

계획

- 체계적이고 명확한 계획을 위한 6가지
 - 문제 정의
 - 타당성 분석
 - 소프트웨어 개발 비용 산정
 - 소프트웨어 개발 비용 산정 기법
 - 일정 계획
 - 위험 분석

부산을 가기 위한 사전 계획

- (1) 어디를 가는가? → 부산 ➡ 문제정의
- (2) 내가 부산을 가는 것이 맞나? 가진 돈은 충분한가? ➡ 타당성 분석
- (3) 숙박비용은 얼마? 교통비용은 얼마? ➡ 소프트웨어 개발 비용 산정, 산정 기법
- (4) 일정은 어떻게 되는가? (3박4일, 4박5일), 아침에 출발? 저녁에 출발? ➡ 일정 계획
- (5) 해수욕장에 해파리는 없을까? 사고가 나지는 않을까? ➡ 위험 분석

계획

- 계획

- 누가 무엇을 어느 기간동안 어떻게 개발해야 하는지 예측하는 작업
- 구현해야 할 기능과 요구하는 성능 및 인터페이스 성능에 따라 개발범위를 정함
- 구현할 프로젝트 특징과 자원 및 생산성에 따라 비용을 예측

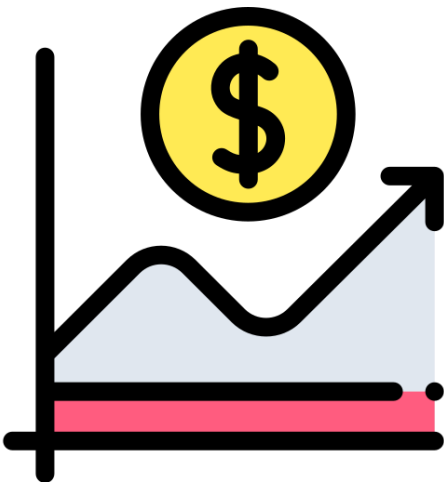
문제 정의

- 문제 정의
 - 소프트웨어 개발의 **첫 작업**
 - 무엇을 개발할 것인지 명확히 정의 (e.g., 모바일 게임, PC 게임)
 - 개발 범위 결정
- 문제 정의 시 필요한 것
 - 개발하고자 하는 영역의 **배경 지식** (e.g., 주식 SW개발 - 차트 등 배경지식)
 - 기존에 존재하는 **유사 시스템을 사용**해보고 분석

타당성 분석

- 타당성 분석

- 개발할 시스템에 대해 **투자 효율성**이 얼마나 높은지, **시장성은 얼마나 큰지** 등을 검토해야 함 (경제적 타당성)
- 사용자가 원하는 수준으로 개발하기 위해 **기술적인 어려움은 없는지** 검토 (기술적 타당성)
- 개발 과정에서 사용하는 프로그램 등에 대해 **소유권 등의 법적 문제는 없는지** (법적 타당성)



경제적 타당성



기술적 타당성



법적 타당성

타당성 분석

- 경제적 타당성

- 경영자: 기업에 얼마나 많은 이윤을 남겨줄 지, 투자 효율성 분석
- 분석가: 투자 대비 효과를 검토 후 경영자에게 정확한 정보 제공
- 시장 분석을 통해 **시장성을 확인**
- **이를 통해 개발 여부를 판단**



- 기술적 타당성

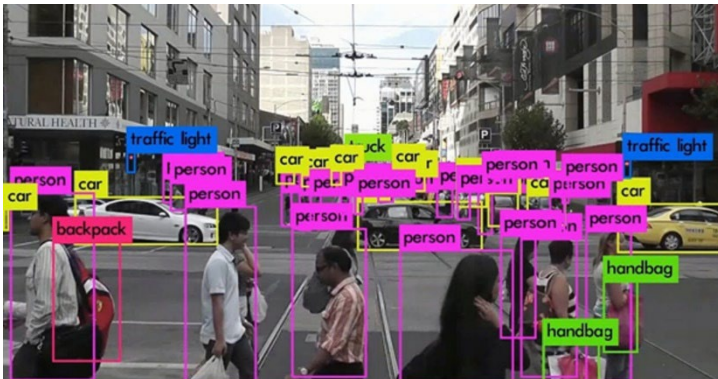
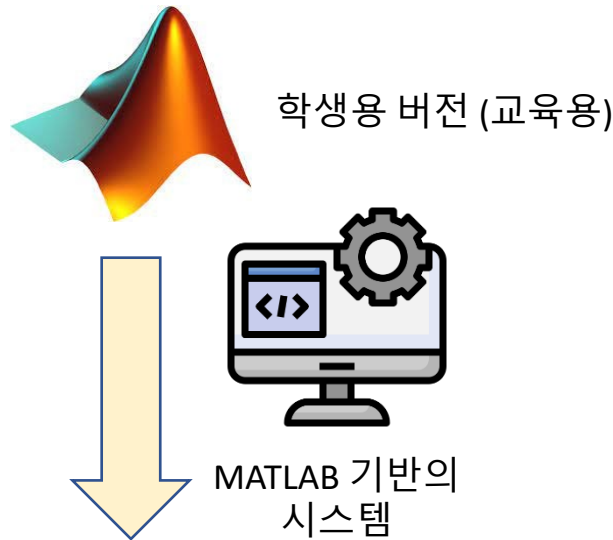
- 현재의 기술로 사용자가 **요구하는 기능을 구현할 수 있는지 검토**
- **하드웨어 성능**이 개발에 지장은 없는지 검토
- **개발자의 기술력**에 문제가 없는지 검토



타당성 분석

- 법적 타당성

- 개발용 소프트웨어와 도구의 사용이 법적으로 문제가 없는지 검토
- 지적 소유권과 프로그램 보호법이 강화 → 법적 문제 꼭 검토해야 함



기업은 개발용 소프트웨어의 도구 사용이 법적으로 문제가 없는지 꼭 검토해야 함

개발 비용 산정

• 개발 비용 산정의 어려움

• 하드웨어

- 제품의 형상이 명확하고 투입되는 부품이 정해짐
- 생산 공정이 표준화, 자동화 → 개발 비용 산정이 쉬움

• 소프트웨어

- 사람(개발자)이 중심이어서 개발 인력의 능력에 따라 품질의 차이가 큼
- 개발 프로세스가 다양해서 표준화나 자동화가 어려움
- 체계적인 개발 비용 산정 방법이 중요함

조립 PC 기본 세트 달기					
CPU	인텔 코어i5-2세대 2500K (샌디브릿지) (정품)	180	1	232,000	삭제
메인보드	ASUS P8P67 PRO (B3) STCOM	80	1	244,000	삭제
RAM	삼성전자 DDR3 4G PC3-10600 (정품)	196	2	24,000	삭제
VGA	MSI 지포스 GTX560 Ti N560GTX-Ti O C D5 1GB 트윈프로저 2	190	1	277,500	삭제
SSD	삼성전자 S470 Series (64GB, MZ-5PA064/KR, 정품)	87	1	140,000	삭제
HDD	WD 500GB Caviar Blue WD5000AAKX (SATA3/7200/16M)	206	1	39,800	삭제
ODD	삼성전자 Super-WriteMaster SH-222AL (블랙 정품벌크)	153	1	21,830	삭제

구분	품목	규격 (피트×인치×인치)	수량(본)	단가(원)	금액(원)
방부목 (Green) 일 경우	기둥	6×6×6	8	22,464	179,712
	동바리목	2×6×6	8	7,488	59,904
	핸드레일류	12×4×2	21	8,463	177,723
	핸드레일류	4×2×2	100	1,274	127,400
	측면/계단판류	12×12×2	13	30,030	390,390
	장선 명에 각재	12×6×2	20	13,104	262,080
	바닥재	12×6×1	120	8,237	988,416
	화이트 래티스	8×4(피트)	7(평)	45,000	315,000
	기타 철물류		10	30,000	300,000
자재비 소계					2,800,625

개발 비용 산정

- 개발 비용에 영향을 주는 요소
 - 프로그래머 자질 (초급 vs. 경험자)
 - 소프트웨어 복잡도 (단순 어플리케이션 개발 vs. 시스템 프로그램 개발)
 - 소프트웨어 크기 (간단한 SW vs. 복잡한 SW)
 - 가용시간
 - 요구되는 신뢰도 수준 (작은 게임에서의 오류 vs. 금융 시스템에서의 오류)
 - 기술 수준 (고급언어(C, C++) 개발자, 저수준(Low-level)언어 개발자)

개발 비용 산정 기법 - 하향식 산정 기법

- 하향식 산정 기법

- 과거의 유사 경험을 바탕으로 회의를 통해 산정
- **전문가 판단 기법**: 조직 내 경험이 있는 2명 이상의 전문가에게 비용 산정
- **델파이 기법**: 1명의 조정자와 다수의 전문가 의견을 종합해 비용 산정

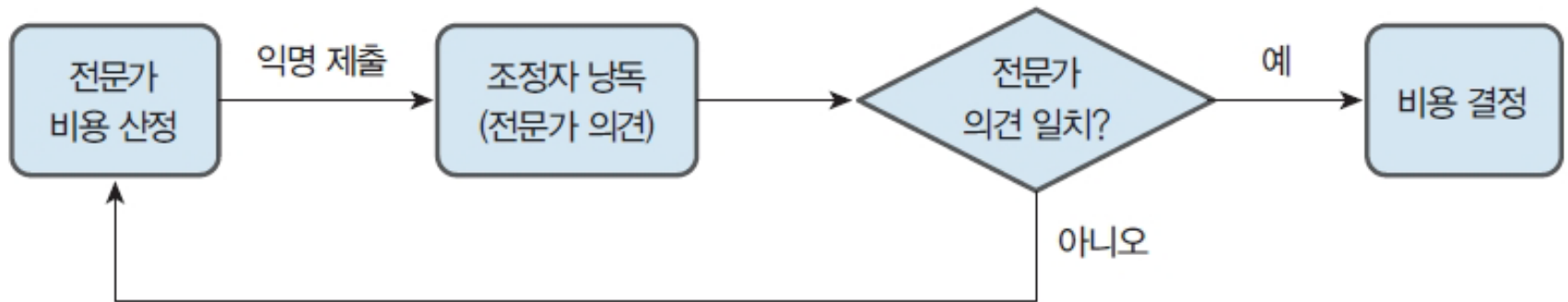
- 전문가 판단 기법

- 소프트웨어 개발 비용을 산정할 때 경험이 많은 전문가들의 의견을 듣고 결정하는 방법
- **장점**
 - **경험이 많은 전문가가 판단을 내림 → 신뢰성 있고 편리함**
 - 짧은 시간에 개발비를 산정하거나 입찰에 응해야 하는 경우
- **단점**
 - 경험에만 의존하는 경우 부정확할 수 있음
 - 새로운 프로젝트도 이전과 비슷하다고 쉽게 생각하여 과소평가
 - **단점보완 → 델파이 기법**

개발 비용 산정 기법 - 하향식 산정 기법

- 델파이 기법

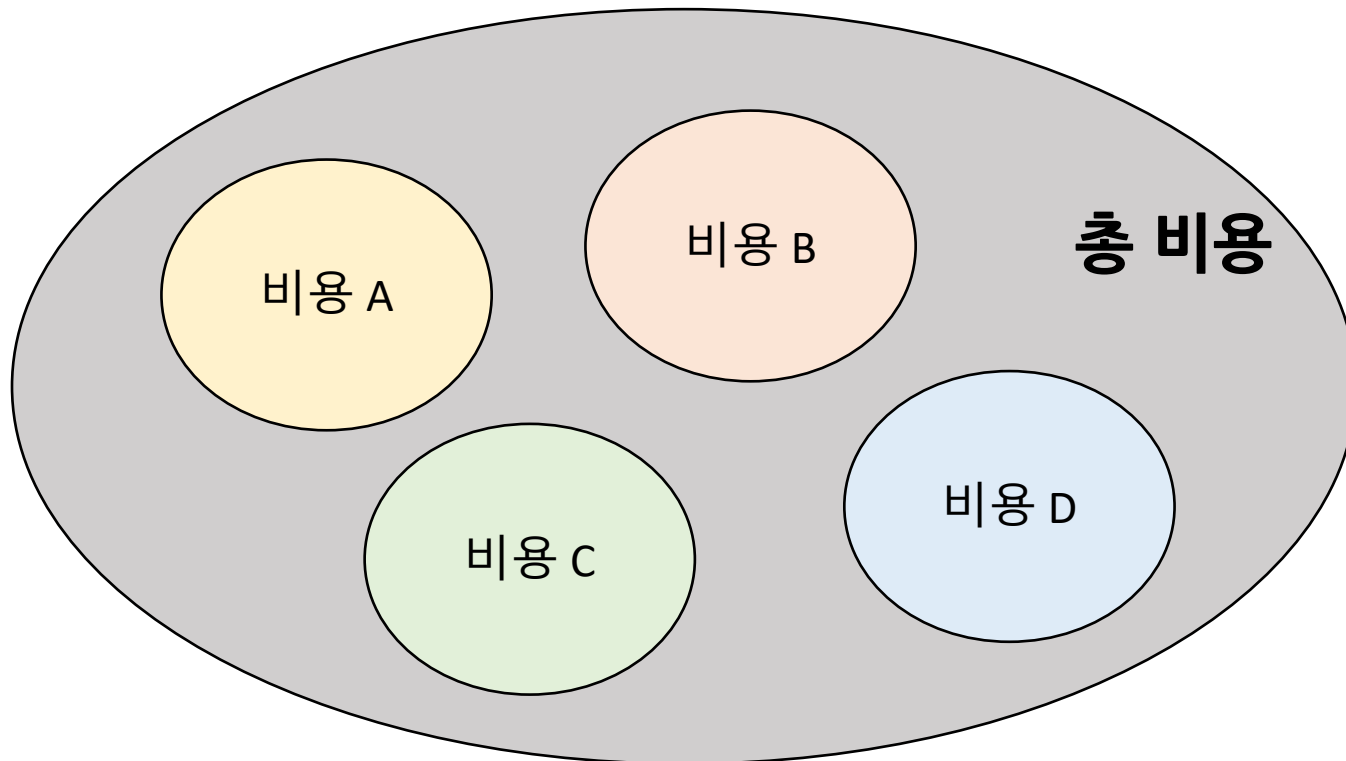
- 전문가의 경험을 중요시하여 비용을 산정하는 것은 전문가 판단 기법과 같음
- **전문자의 편견이나 분위기에 영향을 받지 않도록 조정자를 둠**



개발 비용 산정 기법 - 상향식 산정 기법

- 상향식 산정 기법

- 프로젝트의 세부 작업 단위별로 비용을 산정한 후 전체 비용을 합산하는 방법
- 원시 코드 라인 수 (Line of Code, LOC) 기법
- 개발 단계별 노력 (Effort per Task) 기법



개발 비용 산정 기법 - 상향식 산정 기법

- 원시 코드 라인 수 (**Line of Code, LOC**) 기법
 - 원시 코드 라인 수의 **비관치, 낙관치, 중간치**를 측정 후 **예측치**를 구해 비용 산정

- 낙관치 : 한 모듈의 라인 수를 가장 적게 생각할 때의 예상 라인 수(가중치 1 부여)
- 비관치 : 한 모듈의 라인 수를 가장 많이 생각할 때의 예상 라인 수(가중치 4 부여)
- 중간치 : 한 모듈의 라인 수를 보통이라고 생각할 때의 예상 라인 수(가중치 1 부여)
- 예측 LOC : $(\text{낙관치} + (4 \times \text{중간치}) + \text{비관치}) / 6$

- 예) 낙관치: 300LOC, 비관치: 900 LOC, 중간치: 600 LOC
- 예측 LOC: $(300 + 4 \times 600 + 900) / 6 = 600 \text{ LOC}$

개발 비용 산정 기법 - 상향식 산정 기법

- 개발 단계별 노력 기법

- LOC 기법은 개발 소프트웨어의 총 코드 라인 수를 예측
- But, 실제 소프트웨어 개발에는 **코딩 뿐 아니라 요구분석, 설계** 등에서도 **인력과 자원이 많이 필요**
- **개발 단계별 노력 기법:** 각 기능을 구현하는 데 필요한 **M/M**을 **소프트웨어 개발 생명주기의 각 단계에 적용해 단계별로 산정**
 - M/M: Man Month, **프로젝트에 투입되는 월 인원을 나타내는 숫자**
 - 프로젝트 기간이 1달에 끝내기 위해 필요한 인원
 - 1 M/M (1명이 한달간 하면 끝낼 수 있고, 2명이면 보름)

개발 비용 산정 기법 - 상향식 산정 기법

- 개발 단계별 노력 기법

- 예1) 소프트웨어 개발 기간: 12개월, 5명의 개발자는 12개월, 7명의 개발자가 5개월동안 참여
- 노력 M/M: $(5 \times 12)M/M + (7 \times 5)M/M = 95 M/M$
- 예2) LOC 기법에 의해 예측한 총 라인 50,000, 개발자 10명 참여, 개발자들이 월 평균 500라인 코딩
- 노력 M/M: $50,000 / 500 = 100 M/M$
- 개발 기간: $(100 M/M) / 10명 = 10개월$

개발 비용 산정 기법 - 수학적 산정 기법

- COCOMO (COConstructive COSt MOdel) 방법

$$PM = a \times (KDSI)^b$$

- KDSI (Kilo Delivered Source Information): 소프트웨어의 최종 원시 코드 라인 수 (천 단위)

상수	유형별 값
a	단순형(2.4), 중간형(3.0), 내장형(3.6)
b	단순형(1.05), 중간형(1.12), 내장형(1.20)

- 단순형: 복잡도와 난이도가 비교적 높지 않은 업무용 소프트웨어 (크기: 50KDSI 이하)
- 중간형: 운영체제, 데이터베이스 관리 프로그램 (크기: 300KDSI 이하)
- 내장형: 자동화 기기나 하드웨어와 밀접한 관련이 있는 내장형 소프트웨어 (크기: 300KDSI 이상)

개발 비용 산정 기법 - 수학적 산정 기법

- COCOMO (COConstructive Cost Model) (노력 조정 수치 추가)

$$PM = a \times (KDSI)^b \times EAF$$

- EAF (Effort Adjustment Factor): 노력 조정 수치 (프로젝트의 특징 고려)

특성	비용 승수 요소	승수 값					
		매우 낮음	낮음	정상	높음	매우 높음	극히 높음
제품 특성	요구되는 신뢰도	0.75	0.88	1.00	1.15	1.40	
	데이터베이스 크기	0.94	1.00	1.08	1.16		
	제품의 복잡도	0.70	0.85	1.00	1.15	1.30	1.65
컴퓨터 특성	실행 시간 제약			1.00	1.11	1.30	1.66
	주 기억 장치의 제약			1.00	1.06	1.21	1.56
	HW/SW의 안정성		0.87	1.00	1.15	1.30	
	처리 시간		0.87	1.00	1.07	1.15	
개발자 특성	분석가의 능력	1.46	1.19	1.00	0.86	0.71	
	응용 분야 경험	1.29	1.13	1.00	0.91	0.82	
	컴퓨터와 친숙성	1.21	1.10	1.00	0.90	-	
	프로그래머 능력	1.42	1.17	1.00	0.86	0.70	
	프로그램 언어의 경험	1.14	1.07	1.00	0.95		
프로젝트 특성	소프트웨어 공학 기술 사용	1.24	1.10	1.00	0.91	0.82	
	소프트웨어 도구의 사용	1.24	1.10	1.00	0.91	0.83	
	요구되는 개발 일정	1.23	1.08	1.00	1.04	1.10	

예:

- (1) 요구되는 신뢰도가 높고 (1.15)
- (2) 매우 복잡 (1.30),
- (3) 소프트웨어공학 기술을 거의 사용하지 않음 (1.24),
- (4) 개발 일정이 매우 촉박하고 (1.10),
- (5) 다른 요소들은 보통 (1.00)

$$EAF = 1.15 \times 1.30 \times 1.24 \times 1.10 = 2.04$$

개발 비용 산정 기법 - 수학적 산정 기법

- COCOMO (COConstructive Cost Model) (총 개발 기간 계산)

$$TDEV = 2.5 \times (PM)^c$$

- TDEV (Total DEvelopment time): 총 개발 기간

프로젝트 유형	공식
단순형	$TDEV = 2.5 \times (PM)^{0.38}$
중간형	$TDEV = 2.5 \times (PM)^{0.35}$
내장형	$TDEV = 2.5 \times (PM)^{0.32}$

개발 비용 산정 기법 - 수학적 산정 기법

- 예시

- 개발하려는 KDSI: 60, 소프트웨어 중간형, 노력 조정 수치(EAF): 2.04

$$PM = a \times (KDSI)^b \times EAF$$



$$PM = 3.0 \times (60)^{1.12} \times 2.04 = 600.179$$

- 총 개발 기간은

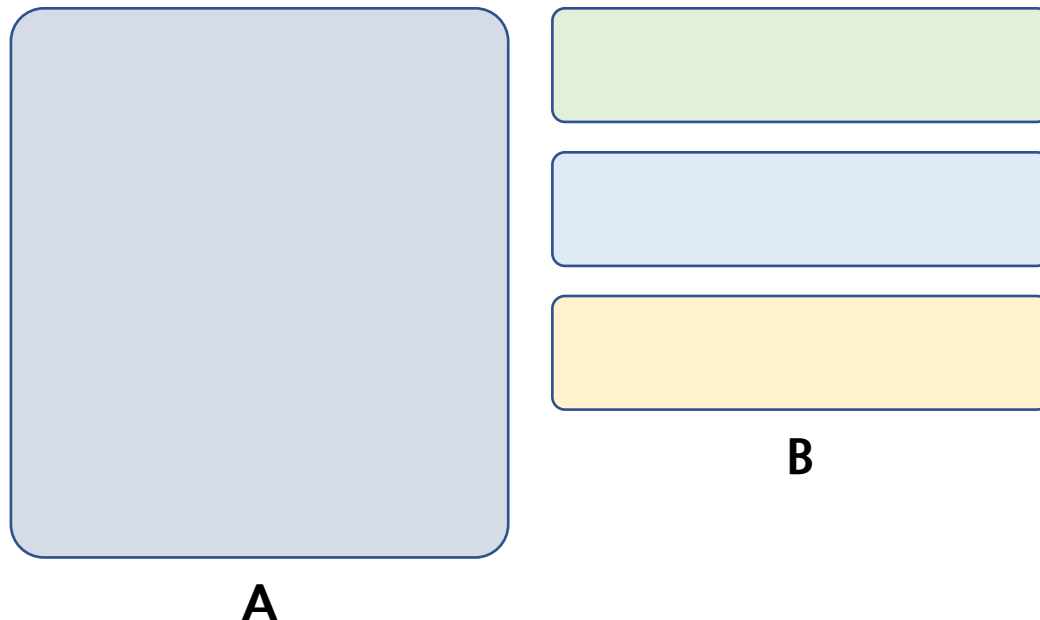
$$TDEV = 2.5 \times (600.179)^{0.35} = 23.461$$

상수	유형별 값	프로젝트 유형	공식
a	단순형(2.4), 중간형(3.0), 내장형(3.6)	단순형	$TDEV = 2.5 \times (PM)^{0.38}$
b	단순형(1.05), 중간형(1.12), 내장형(1.20)	중간형	$TDEV = 2.5 \times (PM)^{0.35}$
		내장형	$TDEV = 2.5 \times (PM)^{0.32}$

개발 비용 산정 기법 - 수학적 산정 기법

- 기능 점수 산정 방법

- **COCOMO 방법의 한계: 계획 단계**에서 원시 코드의 라인 수를 정확히 예측할 수 없음
- **언어의 종류, 개발자의 수준, 알고리즘**에 따라 생성되는 코드의 라인 수가 달라짐
- 라인 수와 무관하게 기능이 많으면 규모도 크고 복잡도도 높다고 판단



COCOMO 방법: 비용 $A > B$





기능 점수 산정 방법: 비용 $A < B$

개발 비용 산정 기법 - 수학적 산정 기법

- 기능 점수 산정 방법
 - 기능 수 (함수의 입출력 수)를 고려



• 프로그래밍 언어 종류

1	Java		11	MATLAB	
2	C		12	R	
3	Python		13	Perl	
4	C++		14	Assembly Language	
5	Visual Basic .NET		15	Swift	
6	Javascript		16	Go	
7	C#		17	Delphi/Object Pascal	
8	PHP		18	Ruby	
9	SQL		19	PL/SQL	
10	Objective-C		20	Visual Basic	

개발 비용 산정 기법 - 수학적 산정 기법

• 기능 점수 산정 방법 (장점)

- 사용자의 요구사항만으로 기능을 추출해 측정
 - 구현 기술, 구현 언어, 개발자의 능력에 상관없이 소프트웨어의 규모를 일관성 있게 제공
- 객관적인 요구사항만으로 측정

• 기능 점수 산정 방법 (단점)

- 높은 분석 능력 필요
 - 요구사항으로부터 기능을 도출하기 위해 **분석 능력**이 필요
- 이 방법을 잘 도출할 수 있는 기능 점수 전문가 필요
- 개발 규모를 예측하는 데는 적합하지만, **실제 개발과는 다를 수 있음**

Questions?