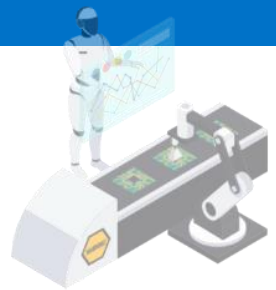
3강. 프로젝트 관리

컴퓨터과학과 김희천 교수

목차



Chapter. 1

프로젝트 관리 개요

1. 프로젝트 관리의 필요성

- 예산과 일정의 제약으로 관리가 필요
- 프로젝트 관리란 프로젝트를 계획하고 감독하는 일
  - 계획의 수립
  - 고객의 요구, 지켜야 하는 표준을 따르는지 확인
  - 시간과 예산에 맞추어 개발되는지 사람과 프로세스를 제어
- 소프트웨어 프로젝트 관리가 어려운 점
  - 진척 관리를 위해 문서에 의존함
  - 소프트웨어 개발 프로세스에 관한 명확한 표준이 없음
  - 기술 발전 속도가 빨라 프로젝트 경험을 살리기 어려움

2. 프로젝트 관리자의 업무

프로젝트 착수

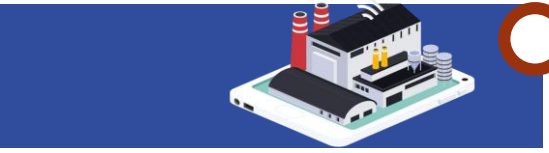
- 제안서의 작성

프로젝트 계획

- 일정, 비용, 자원, 위험 계획

프로젝트 실행

- 계획에 기초한 프로젝트 감시와 통제



프로젝트 종료

- 보고서 작성, 프로젝트 평가

3. 프로젝트 계획

- 어떤 일을, 어느 정도의 비용으로, 누구에 의하여, 언제까지 행해져야 하는가를 결정하는 일
- 계획에 기초하여 산출물과 개발 절차를 관리함

브룩스의 법칙

- 가장 흔한 프로젝트 실패는 일정의 지연임
- "일정이 늦어진 소프트웨어 프로젝트에 인력을 추가하는 것은 일정을 더욱 늦추는 결과를 낳는다"
- 기존 업무의 이해, 의사소통 경로의 증가, 작업의 재분할이 필요하기 때문

4. 프로젝트 계획서의 구성

- 개요
- 개발 절차 계획
- 인원, 예산 및 일정 계획
- 문서화 계획
- 하드웨어와 소프트웨어 자원 계획
- 위험 관리 계획

Chapter. 2

소프트웨어 일정 계획

1. 일정 계획과 작업 (1/2)

- 일정계획은 작업의 분할, 작업들 간의 관계, 인적/물적 자원의 배정 등을 계획하는 것
- 작업의 분할
  - 요구명세서에 기초하여 전체 작업을 관리 가능하고 측정 가능한 소작업들로 분할
  - 작업 분해 구조(WBS)
- 작업의 명세화
  - 소작업에 대해 일의 양, 필요한 산출물과 컴퓨터 자원 등을 결정
  - 작업의 양을 인원-월(PM)로 표시함, 1PM은 중급 수준 개발자의 한 달간 작업량

1. 일정 계획과 작업 (2/2)

- **작업 진행 순서의 정의**
  - 작업들의 선후 관계를 분석하여 순서를 정함
  - PERT/CPM
- **인력 배정**
  - 작업의 양과 특성에 맞도록 개발자를 배정
- **작업 비용의 산정**
  - 작업의 양과 인력에 따른 비용을 산정
- **개발 일정의 수립**
  - 작업별로 시작 시점과 종료 시점을 설정
  - CPM으로 분석하고 간트(Gantt) 차트로 도표화

2. WBS

- 작업 분할 구조(Work Breakdown Structure)
  - 프로젝트 수행을 위해 개발 업무를 분할하여 계층 구조로 표현
- 최하위 수준의 작업을 작업 패키지(work package)라고 함
  - 비용이나 기간의 정량적 측정이 가능한 입력물과 출력물을 가짐
- 프로젝트 계획과 관리를 위한 기초 자료

3. PERT와 CPM

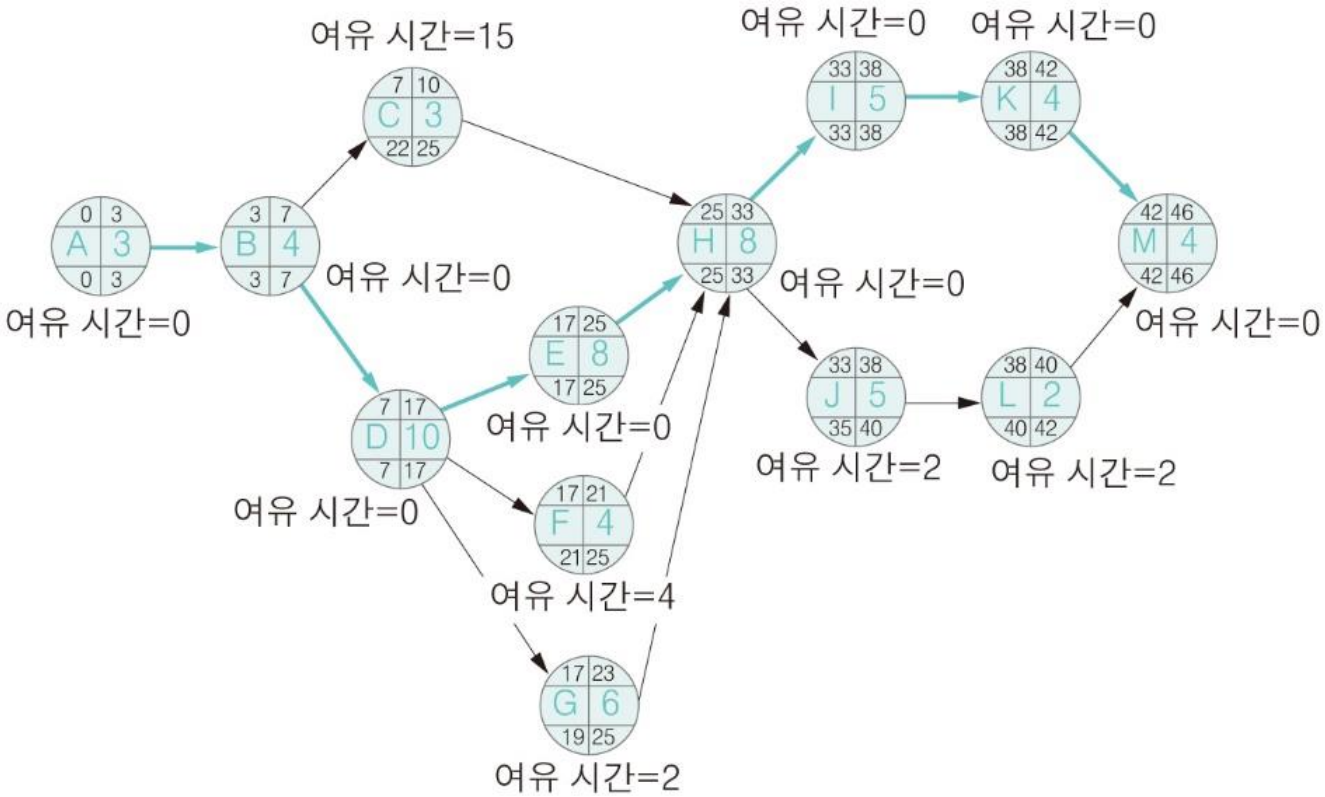
PERT

- 작업들의 선후 관계를 표현한 사이클이 없는 방향 그래프

CPM - 임계 경로 방법

- 일정 계획을 위한 알고리즘적 분석 방법
- 임계 경로는 시작에서 종료 작업까지의 경로 중 가장 긴 경로
- 임계 경로상의 작업들은 프로젝트의 일정 준수를 위해 지연이 허용되지 않는 작업
- 임계 경로상에 있지 않은 작업들은 여유 시간을 가짐

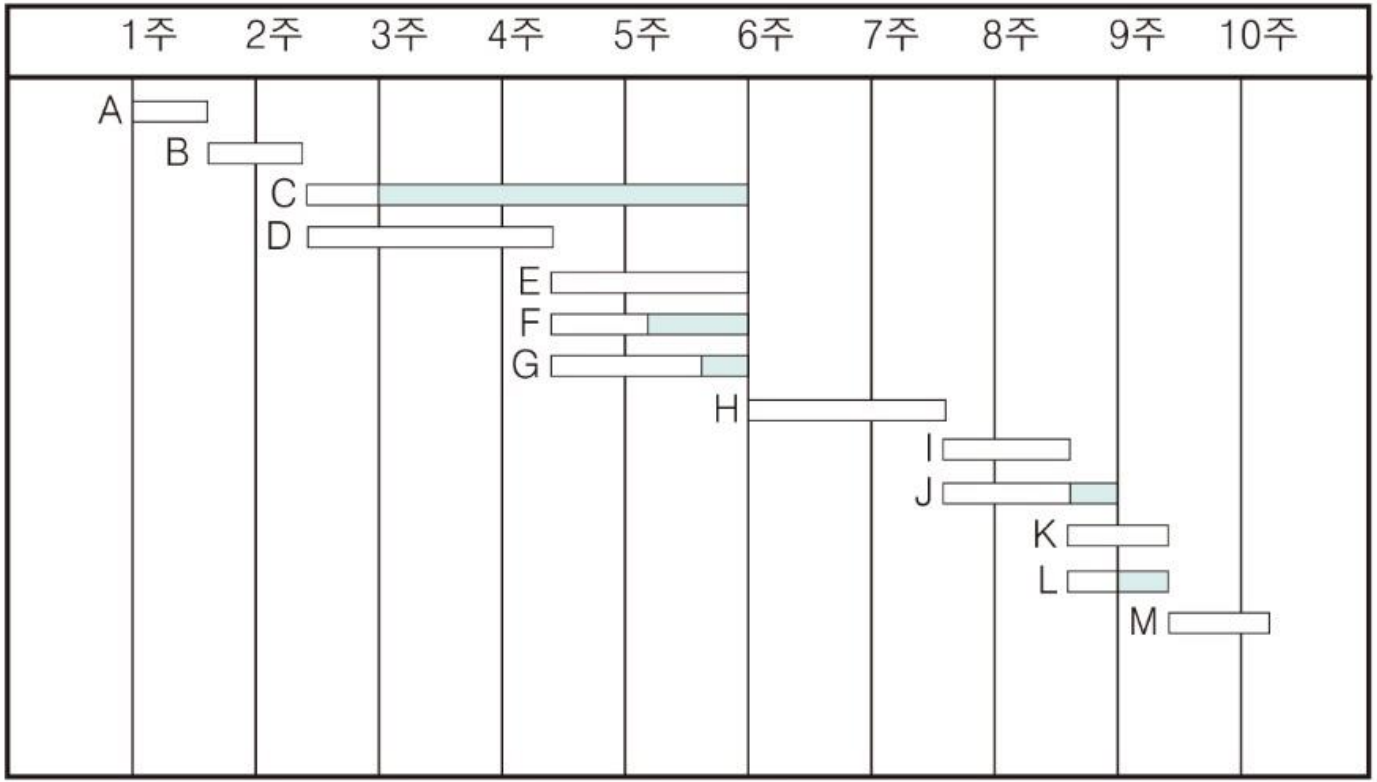
4. CPM 네트워크


여유 시간=15
여유 시간=0
여유 시간=0
0 3
A 3
0 3
여유 시간=0
3 7
B 4
3 7
여유 시간=0
7 10
C 3
22 25
25 33
H 8
25 33
여유 시간=0
33 38
I 5
33 38
38 42
K 4
38 42
42 46
M 4
42 46
여유 시간=0
17 25
E 8
17 25
여유 시간=0
7 17
D 10
7 17
여유 시간=0
33 38
J 5
35 40
여유 시간=2
38 40
L 2
40 42
여유 시간=2
17 21
F 4
21 25
여유 시간=4
17 23
G 6
19 25
여유 시간=2

5. 간트 차트 (1/2)

- 막대 모양으로 프로젝트 작업들의 순차 또는 병행 순서를 보여주는 차트
  - 상단에 시간축을 표시, 작업별로 막대를 가로방향으로 표시
  - 막대는 작업 시간에 맞추어지며, 길이는 소요 시간을 의미
- 일정 조정, 인력 배정 계획에 사용됨
  - 작업별로 진척 비율이나 인력 배정을 표시할 수 있음

5. 간트 차트 (2/2)



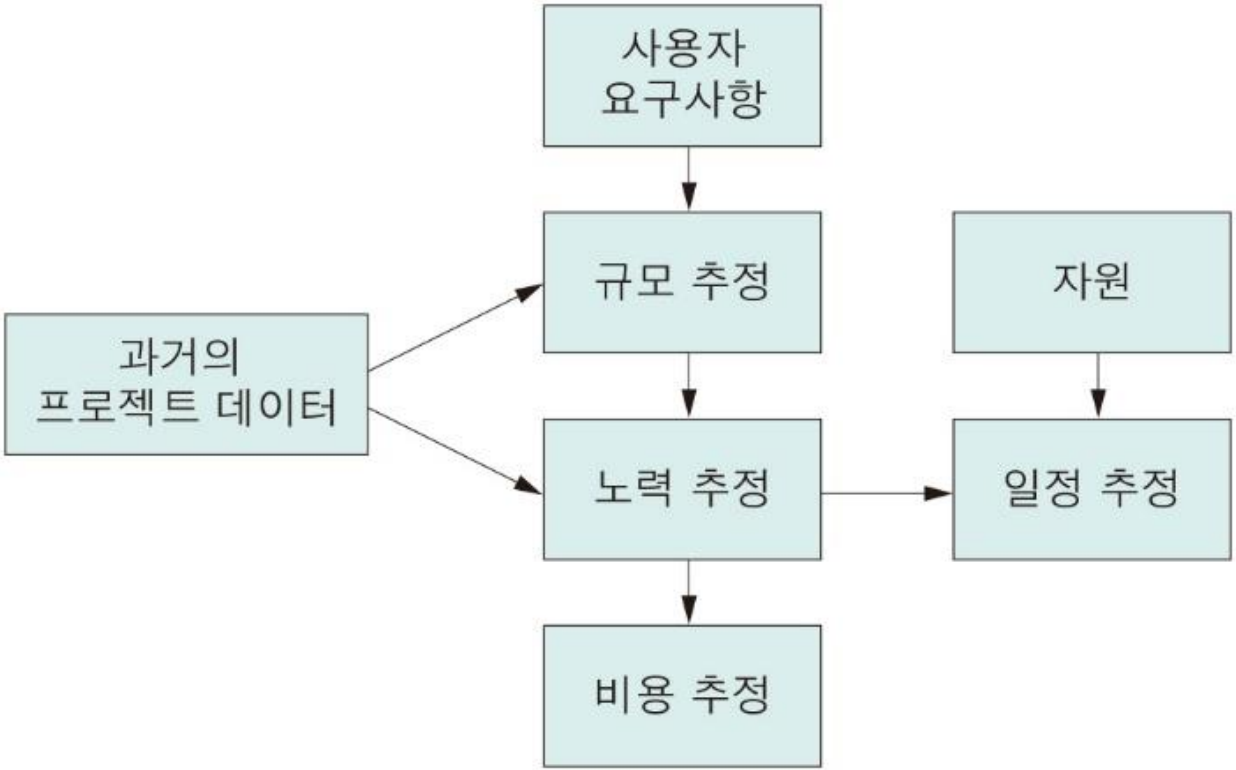
Chapter. 3

소프트웨어 규모의 산정

1. 소프트웨어 프로젝트 산정 (1/2)

- 규모 추정→ 개발 노력(비용) 추정, 일정 계획
- 규모 산정의 정확성이 다른 부분에 영향을 줌
  - 고객 요구사항과 시스템 명세서를 참고
  - 라인수(LOC)를 추정하는 방법, 기능 점수(FP) 방법
- 노력 추정에 과거 데이터와 프로젝트의 특성을 고려
- 일정 계획에는 투입 되는 인력을 고려
- 추정의 정확성은 과거 프로젝트 데이터의 정확성, 제공되는 입력의 정확성, 개발 조직에서 프로세스의 성숙도에 좌우됨

1. 소프트웨어 프로젝트 산정 (2/2)



2. 기능 점수 (1/3)

- 기능 점수는 기능의 규모를 측정하기 위한 단위
  - 소프트웨어의 기능을 다섯 가지 유형으로 분류하여 계산함
- 프로그램의 기능에 초점을 맞춘 논리적 규모 척도
  - 기능적 사용자 요구사항을 양으로 표시
- 구현 기술이나 구현 언어와 무관
- 사무 정보 시스템의 규모 산정에 적합함
- 보정 기능 점수(AFP)는 미보정 기능 점수(UFP)와 보정 계수(VAF)의 곱

2. 기능 점수 (2/3)

✚ 미보정 기능 점수 (UFP)

- 프로그램에서 표현되거나 사용되는 데이터의 총량을 계량화
- 데이터 기능(내부 논리 파일, 외부 인터페이스 파일)과 트랜잭션 기능(외부 입력, 외부 조회, 외부 출력)의 개수를 측정
- 각각에 복잡도에 따른 가중치를 곱하여 합산함

✕ 보정 기능 점수 (AFP)

- AFP = UFP*VAF, 여기서 VAF = 0.65 + 0.01*TDI(총 영향도)
- VAF는 보정 계수이며(0.65~1.35), TDI는 기술적 복잡도를 반영하기 위해 14개 항목의 영향도(0~5)를 모두 합한 것(0~70)

2. 기능 점수 (3/3)

표 3-1 | UFP 계산표 양식

기능 유형		빈도수	정보처리 기능 수준			합
			간단	보통	복잡	
데이터 기능	내부 논리 파일(ILF)		__×7	__×10	__×15	
	외부 인터페이스 파일 (EIF)		__×5	__×7	__×10	
트랜잭션 기능	외부 입력(EI)		__×3	__×4	__×6	
	외부 출력(EO)		__×4	__×5	__×7	
	외부 조회(EQ)		__×3	__×4	__×6	
미보정 기능 점수						

표 3-2 | 복잡도 항목과 영향도

	특성	영향도		특성	영향도
1	데이터 통신		8	온라인 변경	
2	분산 데이터 처리		9	처리 복잡도	
3	성능		10	재사용성	
4	과부하 컴퓨터의 사용		11	설치 용이성	
5	트랜잭션 비율		12	운영 용이성	
6	온라인 데이터 입력		13	다중 설치성	
7	사용자 효율		14	변경 용이성	
총 영향도					

3. 기능 점수 고찰

- 프로그래밍 언어별로 LOC/FP 즉, 기능 점수 1점을 구현하기 위해 필요한 라인 수가 존재
- 기능 점수로부터 라인 수를 계산할 수 있음
- 초기 단계에서 라인 수 추정에 효과적
- 프로그래머의 평균 생산성(FP/PM)을 안다면, 전체 PM을 계산할 수 있음
- 사무 정보 시스템의 규모 산정에 적합함
- 사례는 교재 참고

Chapter. 4

소프트웨어 개발 비용 산정

1. 비용 산정 방법의 분류

+ 판단에 의한 방법
  - 전문가의 판단, 델파이 방법, 작업 분해에 의한 방법
+ 수학적 모델을 이용한 방법
  - 알고리즘 모델, 유추에 의한 산정

COCOMO

- 가장 잘 알려진 소프트웨어 비용 산정 모델(COnstructive COst Model)
- 프로젝트 유형을 3가지로 구분(기본/중간/내장형)
- 분석 정도에 따른 3가지 모델을 제시(기본/중급/상세)

2. COCOMO

- 효과적 산정을 위해 먼저 규모를 추정해야 함
- 기본 COCOMO는 라인 수만으로 비용을 추정함
  - 대략적으로 개발 노력은 소프트웨어 규모에 선형적으로 비례

라인수(LOC)

- 간단하며 비용 산정 방법과의 연결이 용이하고 직관적임
- 계획단계에서 산정하기 어렵고 프로그래밍 언어에 따라 다름
- 과거의 경험, 전문가의 판단, 구성 요소별 산출한 후 합산

3. 기본 COCOMO (1/2)

✚ 프로젝트 유형

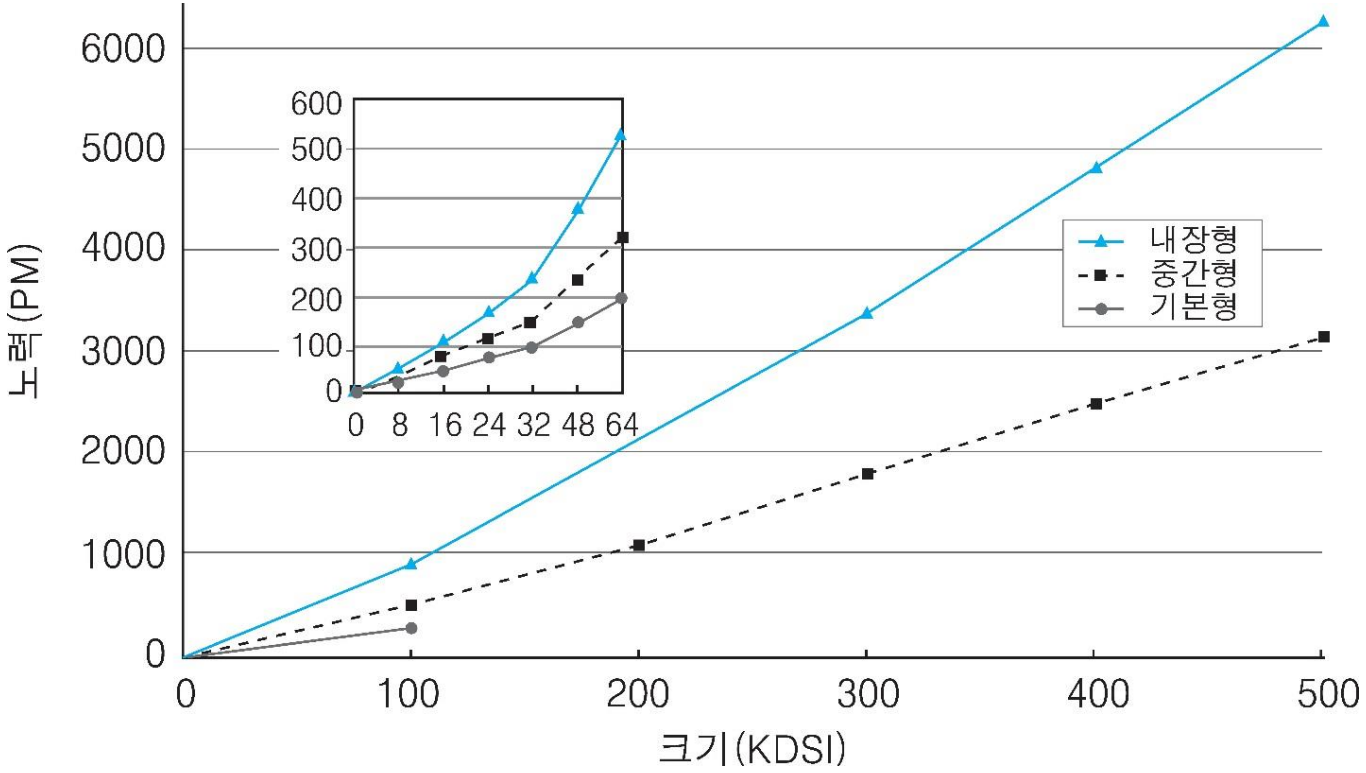
- ✕기본형 : 소규모 프로젝트(~50KDSI), 경험 있는 개발자, 까다롭지 않은 요구사항
- ✕중간형 : 중규모 프로젝트 (~300KDSI), 중간 정도의 경험, 요구사항의 혼재
- ✕내장형 : 대규모 프로젝트, 제한된 하드웨어, 엄격한 운영 조건

	총 노력(PM)	총 개발 기간(개월)
기본형	$Effort=(2.4*KLOC^{1.05})$	$Tdev=2.5*(Effort)^{0.38}$
중간형	$Effort=(3.0*KLOC^{1.12})$	$Tdev=2.5*(Effort)^{0.35}$
내장형	$Effort=(3.6*KLOC^{1.20})$	$Tdev=2.5*(Effort)^{0.32}$

3. 기본 COCOMO (2/2)

- 32,000 LOC 기본형 소프트웨어의 추정 예
  - 총 노력 = $2.4 * 32^{1.05} = 91$ PM
  - 총 개발 기간 = $2.5*(91)^{0.38} = 14$ 개월
  - 평균 인력 = 91/14 = 6.5 명
  - 생산성 = 32,000/91 = 352 LOC/PM

4. 기본 COCOMO - 프로젝트 크기와 노력



5. 중급 COCOMO

- 15개의 비용 승수를 곱하여 노력 보정 계수(EAF)를 계산
- 각 비용 승수는 6개의 등급으로 나뉨
- 총 노력을 계산할 때 EAF를 곱함

	총 노력(PM)	총 개발 기간(개월)
기본형	$Effort=(3.2*KLOC^{1.05})*EAF$	$Tdev=2.5*(Effort)^{0.38}$
중간형	$Effort=(3.0*KLOC^{1.12})*EAF$	$Tdev=2.5*(Effort)^{0.35}$
내장형	$Effort=(2.8*KLOC^{1.20})*EAF$	$Tdev=2.5*(Effort)^{0.32}$

6. COCOMO 비용 승수 (1/2)

승 수	비용 승수값					
	매우낮음	낮음	보통	높음	매우높음	극히높음
제품의 속성						
요구되는 S/W 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 속성						
실행 시간의 제약			1.00	1.11	1.30	1.66
주기억 장치의 제약			1.00	1.06	1.21	1.56
컴퓨터와 S/W의 안정성		0.87	1.00	1.15	1.30	
요구되는 응답 시간		0.87	1.00	1.07	1.15	

6. COCOMO 비용 승수 (2/2)

승 수	비용 승수값					
	매우낮음	낮음	보통	높음	매우높음	극히높음
개발 요원의 속성						
분석가의 능력	1.46	1.19	1.00	0.86	0.71	
응용에 대한 경험	1.29	1.13	1.00	0.91	0.82	
엔지니어의 능력	1.46	1.17	1.00	0.86	0.70	
컴퓨터와 S/W의 친숙도	1.21	1.10	1.00	0.90		
프로그래밍 언어 경험	1.14	1.07	1.00	0.95		
프로젝트 속성						
소프트웨어 공학의 활용	1.24	1.10	1.00	0.91	0.82	
도구의 사용	1.24	1.10	1.00	0.91	0.83	
요구되는 개발 일정	1.23	1.08	1.00	1.10	1.10	
예: 산출된 노력 보정 계수 ≒ 1.17						

7. 중급 COCOMO 예

- 4,000 LOC 내장형 소프트웨어의 추정 예
  - 보정 계수는 1.17로 가정
  - 총 노력 = $2.8 * 4^{1.20} * 1.17 = 17$ PM
  - 총 개발 기간 = $2.5 * 17^{0.32} = 6$ 개월
  - 비용 = 17 PM * 5,000,000원/PM = 850,000,000원

8. 소프트웨어의 수정을 위한 노력

- 설계, 코드, 통합과 테스트 부분에서 수정이 필요한 비율을 구하여 수정 보정 계수(AAF)를 계산함
- AAF = 0.4*(설계 수정 비율) + 0.3*(코드 수정 비율) + 0.3*(통합과 테스트 수정 비율)
  - 전체에서 수정이 요구되는 비율을 의미함
- 상응 LOC = 기존 LOC * AAF
  - AAF를 이용하여 수정이 요구되는 LOC를 계산한 후 공식에 대입

9. 기능 점수에 기초한 개발비 산정 사례

- 총 개발 비용은 (보정 후 개발원가 + 이윤 + 경비)
- 보정 후 개발원가는 (기능 점수*기능 점수당 단가)* 보정 계수
- 사례
  - 보정 후 개발 원가=516.5FP * 1.1223 * 519,203원/FP=300,965,338원
  - 총 개발 비용=
    300,965,338원+(이윤)27,086,880원+(경비)10,000,000원≒3억4천만원

Chapter. 5

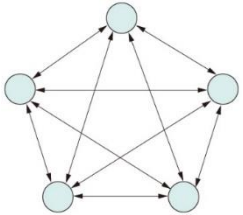
팀 구성 방식

1. 매트릭스 조직 (1/2)

- 프로젝트 조직과 기능별 조직의 장점을 조합한 형태
  - 프로젝트 조직은 프로젝트 동안 유지되는 팀
  - 기능별 조직은 분석/설계/구현 등 기능별로 전문화된 팀
- 개발자가 전문 기능 부서에 속하되, 일정 기간 프로젝트에 소속되는 형태
- 팀 구성원들 간에 정보와 경험을 공유할 수 있으나 기능 부서 관리자와 프로젝트 관리자 양쪽의 지배를 받음

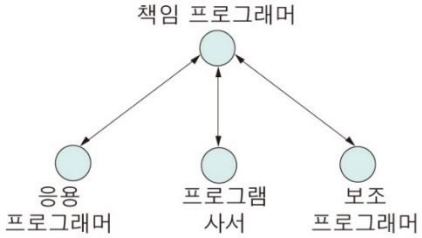
1. 매트릭스 조직 (2/2)

2. 의사 결정 방법에 따른 팀 구성



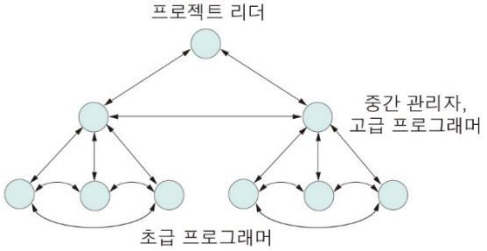
비이기적인 팀

- 분산 형 팀 구성 방식
- 구성원 전체가 의사 결정에 참여
- 작업에 대한 만족도가 높으나 의사 결정이 늦고 책임 소재가 모호함



책임 프로그래머 팀

- 중앙 집중형 팀 구성 방식
- 책임 프로그래머가 중요한 기술적 판단과 관리적 결정을 함
- 의사소통 경로가 감소하나 책임 프로그래머의 능력에 크게 의존함



계층형 팀

- 중앙 집중형과 분산형 팀 구성 방법을 혼합

Chapter. 6

위험 분석과 관리

1. 위험

위험은 불확실성으로 인해 잠재해 있는 문제

- 불분명한 요구사항, 요구의 변경, 추정의 어려움 등 불확실성이 존재
- 위험이 발생하면 제품의 품질, 프로젝트 일정이나 비용, 조직 등에 부정적 영향을 줌

위험 관리

- 가능한 위험 요인들을 예측하고 발생 가능성과 영향력을 분석하여 대책을 계획하고 위험을 관리하는 것

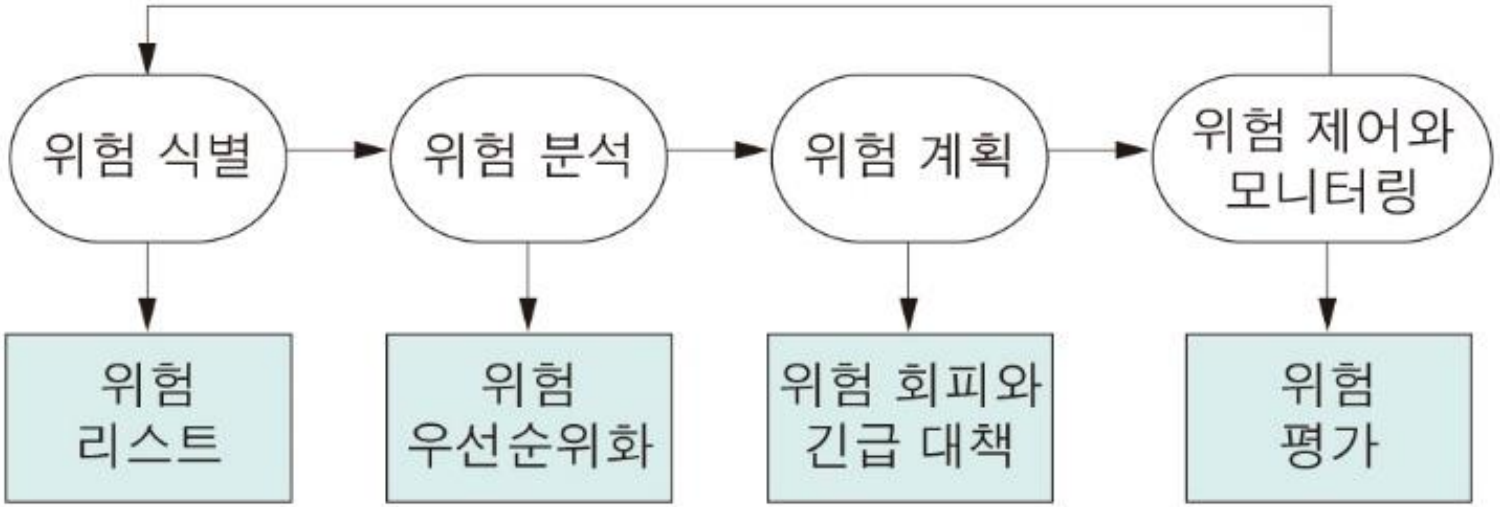
2. 위험의 분류

- 제품의 품질이나 성능에 영향을 주는 위험
- 불안정한 요구사항이나 성능이 낮은 도구 등

- 조직의 비즈니스에 영향을 주는 위험
- 기술의 변화나 경쟁사 제품의 출시 등

- 프로젝트 일정이나 자원 활용에 영향을 주는 위험
- 미흡한 조직의 지원, 중요 프로젝트 요원의 이직 등

3. 위험 관리 프로세스 (1/2)



3. 위험 관리 프로세스 (2/2)

- **위험 식별**
  - 발생 가능한 위험 요인을 나열
- **위험 분석**
  - 위험 요인 별로 발생 가능성과 결과의 심각성을 평가
  - 우선순위를 정하여 대책이 필요한 위험 요인들을 정리
- **위험 계획**
  - 회피 전략 : 발생 가능성을 줄이는 것
  - 최소화 전략 : 위험 발생시 충격을 줄이는 것
  - 긴급 대책 : 최악의 상황에 대비하는 것
- **위험 제어와 모니터링**

다음강의

4강. 소프트웨어 품질