



## 2장. 프로젝트 관리



# Contents

1. 프로젝트 관리의 개념
2. 프로젝트 관리 기법
3. 프로젝트 조직
4. 프로젝트 관리 계획서

# SW개발 프로젝트 실패 원인

- 부족한 소프트웨어 마인드

- 소프트웨어는 하드웨어와 달리 물리적으로 존재하지 않기 때문에 언제든지 변경 가능하다?

- 소프트웨어 공학기술의 활용 미흡

- 업무 증가와 일정 지연 등의 이유로 공학 기법 적용을 미룸

- 부족한 프로젝트 관리 기술

- 소프트웨어 개발 기술의 고도화, 다양한 응용  
도메인으로 프로젝트 수행 시 고려해야할 요소가  
기하급수적으로 증가

# 1. 프로젝트 관리(management)

- SW 개발 프로젝트의 목표
  - 최소의 비용으로 최고의 품질을 유지하는 SW를 성공적으로 개발하는 것
- 프로젝트 관리의 목적
  - 작업 수행에 필요한 여러 가지 자원, 인력, 비용, 재료, 기술 등을 가장 효과적으로 사용하여 프로젝트의 목표를 달성하는 것
- SW개발 프로젝트 관리를 어렵게 하는 요인
  - 보이지 않는 소프트웨어, 빠르게 변하는 기술, 조직마다 다른 프로세스

# 1.1 프로젝트 관리 단계

## 단계 1: 계획 수립

- 프로젝트 목표의 이해와 문서화
- 스케줄, 예산, 기타 자원 요구사항 등 개발

## 단계 2: 자원 획득

- 공간, 컴퓨터 자원, 관련 자료 및 인적 자원의 확보

## 단계 3: 실행

- 계획의 이행

## 단계 4: 모니터링

- 프로젝트 진척도 체크
- 계획과의 편차를 해결하는 대책 강구

## 1.2 프로젝트 계획

- 목표 설정 : 달성 목표와 수행되어야 할 기본 작업(WBS), 산출 결과물과 승인 조건, 가정과 제약조건, 산출물과 점검 일정
- 일정 계획(Scheduling) : 개발 프로세스를 이루는 소작업(activity)를 파악하고 순서와 일정을 정하는 작업
- 비용 추정 : 소프트웨어 개발 비용 예측(estimation)
  - 정확한 비용 예측은 매우 어려움
  - 알려지지 않은 요소가 산재하고 원가 계산이 어려움

시간 \* 인건비

## 2. 프로젝트 관리 기법

- 일정관리
  - WBS와 PERT, 간트차트
- 비용관리
  - 비용의 예측은 어렵기 때문에 작업이 완료되기 위해 필요한 노동의 양인 노력(Effort)의 값으로 추정
  - COCOMO, 기능점수 산정법, 전문가 판단 등
- 위험관리

국제표준

## 2.1 일정 관리 기법 (1/3)

- 요구사항의 복잡도, 개발자 규모 및 기술 수준, 팀 구성 형태, 개발 중에 발생 가능한 리스크 정도 등을 고려하여 계획
- 기본 입력 사항
  - 어떤 프로세스 모델을 선정하였는가
  - 어떤 베이스라인(baseline)을 정의할 것인가
  - 베이스라인 : 어떤 변경이 발생했을 때 모든 관련자의 합의를 거쳐 변경 사항을 결정한 다음 후속 작업을 수행하기 위한 출발점 (버전관리)



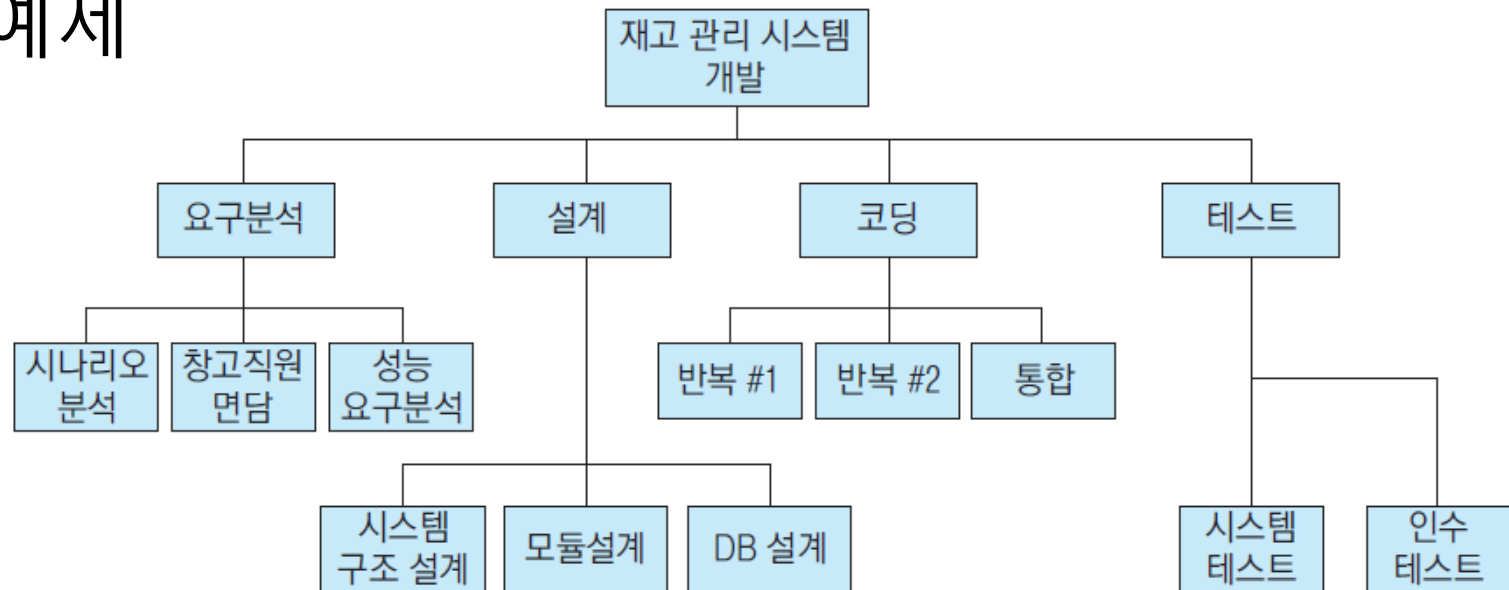
## 2.1 일정 관리 기법 (2/3)

### ● 작업 분할도(WBS: Work Breakdown Structure)

- 프로젝트의 전체 목표를 중간의 세부 목표들로 쪼개어 나타낸 작업 목록
- 프로젝트에서 수행해야 하는 모든 종류의 작업을 식별해서 트리(Tree) 형태로 표현
- 아주 작은 단위로  
나눔! 식별된 작업(task): 추정 가능한 양의 작업,  
독립적으로 이루어질 수 있다고 판단될 때까지 분할
- 엑셀, MS Project와 같은 도구를 사용하여 작성
- 문서화 작업, 프로젝트 리뷰와 같은 작업들도 반드시 태스크로 식별

## 2.1 일정 관리 기법 (3/3)

### ● WBS 예제



### ● 일정계획은 WBS를 기초로 하여 정의

1. 작업 사이의 의존 관계 파악
2. PERT를 이용한 여유 시간 계산
3. 소요 자원의 할당

## 2.1.1 PERT/CPM 차트(1/7)

- 불확실한 프로젝트의 일정, 비용 등을 합리적으로 계획하고 관리하는 기법
  - PERT(Project Evaluation & Review Technique)
    - 1958년에 개발
    - 규모가 크고 복잡한 프로젝트를 보다 수월하게 계획할 수 있도록 설계된 프로젝트 관리 체계
  - CPM (Critical Path Method)
    - 프로젝트 전반에 관한 개요를 제공하며, 모든 것을 완료하는 데 필요한 총 시간을 산정
  - CPM 네트워크를 작성하여 전체 프로젝트에 소요되는 시간을 계산

## 2.1.1 PERT/CPM 차트(2/7)

### ● CPM 네트워크

- 작업을 나타내는 노드와 작업 사이의 선후 의존관계를 나타내는 간선으로 구성된 네트워크
- 네트워크의 박스에는 작업의 시작일과 완성일을 표시

시작가능시점

모든작업이 끝나야 시작하면 →

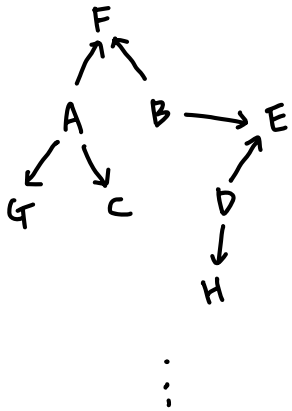
- 가장 이른 시작일(EST, Earliest Starting Time)과 가장 늦은 시작일(LST, Latest Starting Time)을 파악할 수 있음

- CPM 네트워크에서 작업의 여유시간 =  $LST - EST$
- 일부 노드는 이정표(milestones)로 지정 할 수 있음

## 2.1.1 PERT/CPM 차트 (3/7)

- CPM 네트워크 작성을 위한 **소작업 리스트**
  - WBS의 각 작업에 대한 **선후작업관계와 소요시간**

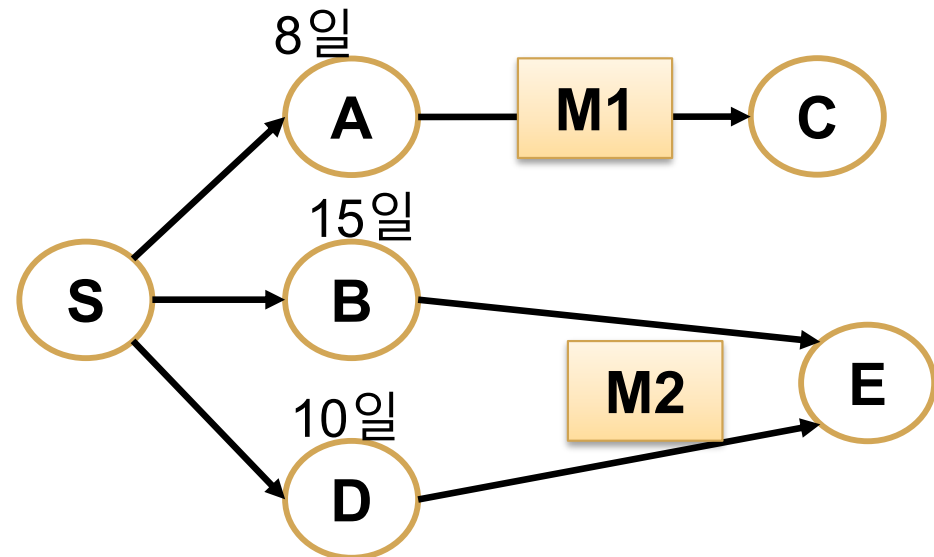
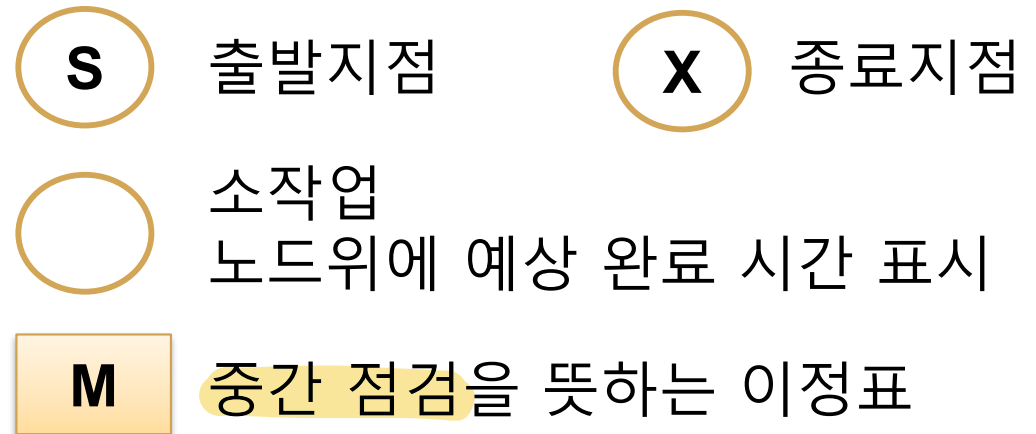
예측



소 작업	선행 작업	소요기간(일)
A	—	8
B	—	15
C	A	15
D	—	10
E	B, D	10
F	A, B	5
G	A	20
H	D	25
I	C, F	15
J	G, E	15
K	I	7
L	K	10

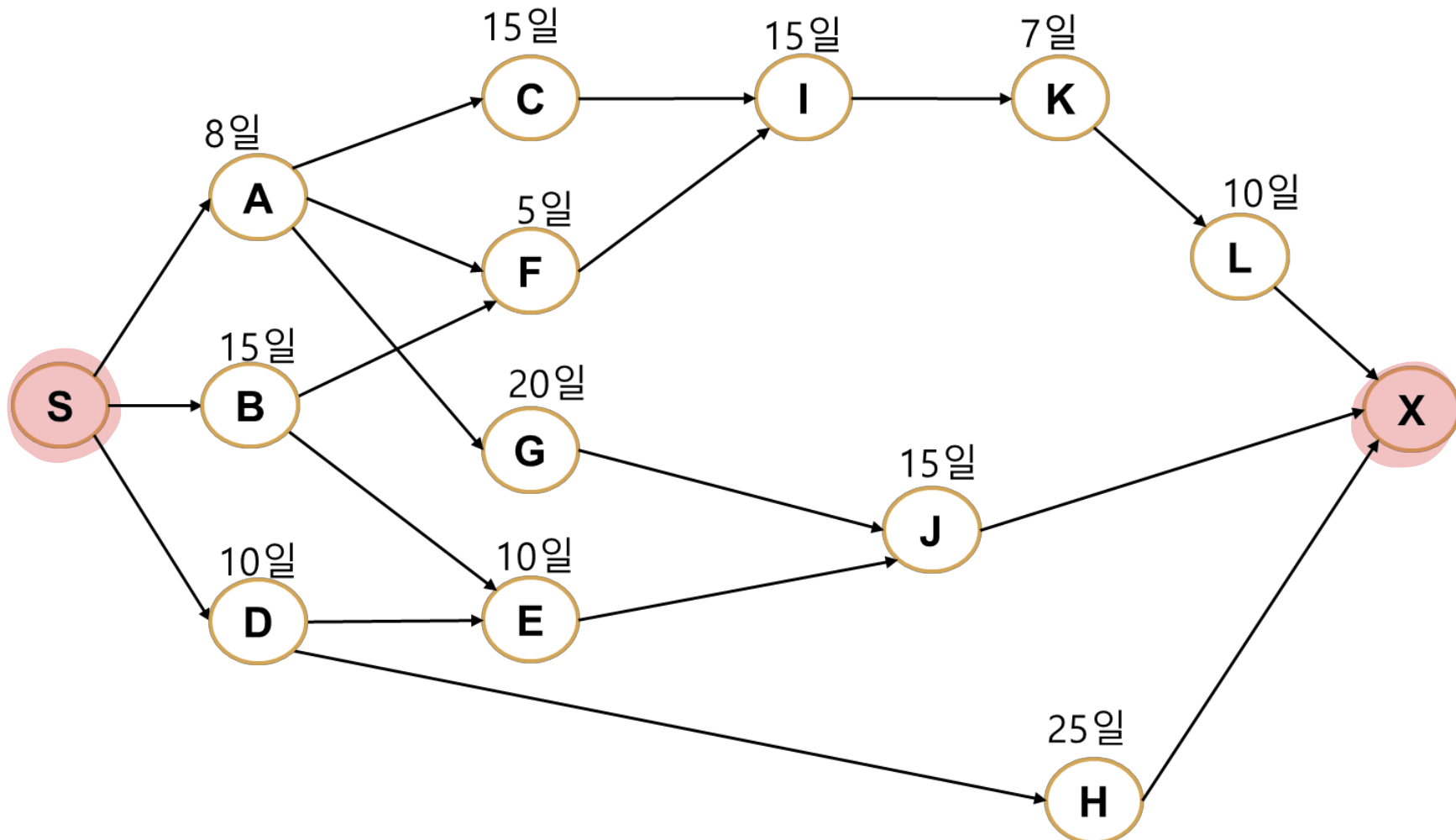
## 2.1.1 PERT/CPM 차트 (4/7)

소 작업	선행 작업	소요 기간 (일)
A	—	8
B	—	15
C	A	15
D	—	10
E	B, D	10
F	A, B	5
G	A	20
H	D	25
I	C, F	15
J	G, E	15
K	I	7
L	K	10

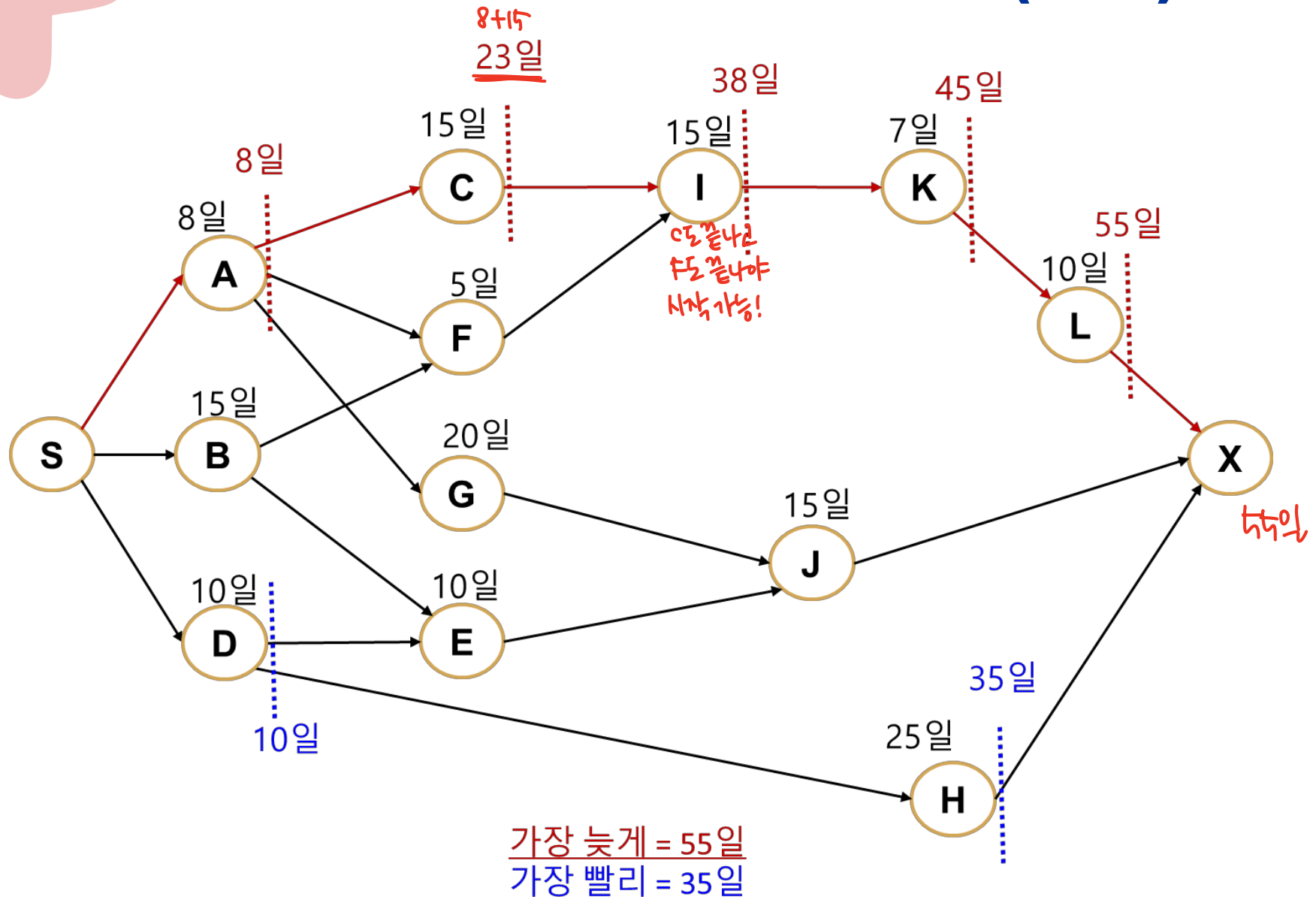


## 2.1.1 PERT/CPM 차트 (5/7)

- 중간점검 없이 표현



# 2.1.1 PERT/CPM 차트 (6/7)





## 2.1.1 PERT/CPM 차트 (7/7)

critical path management

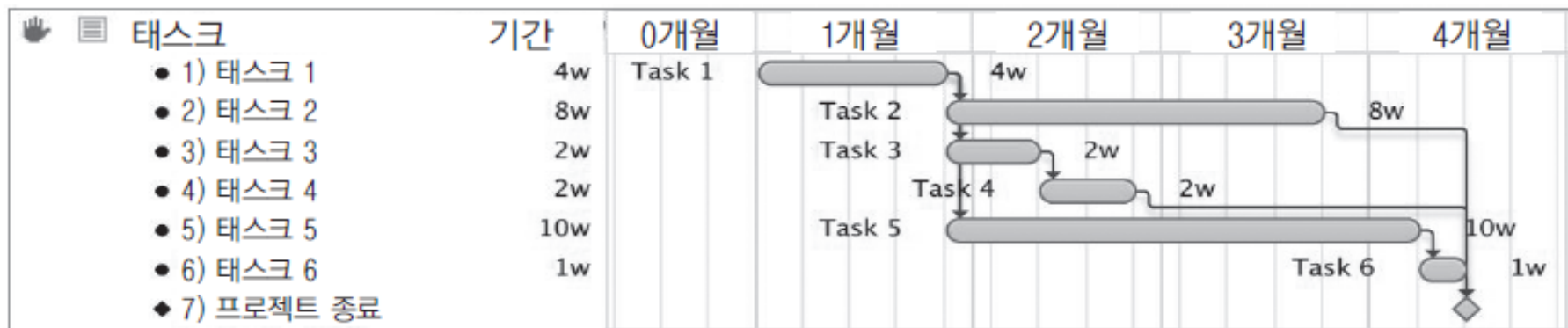
● 임계 경로(critical path) : 가장 소요기간이 긴 경로

가능 경로	소요기간 (일)
S-A-M1-C-M4-I-M6-K-M8-L-X	55
S-A-M3-F-M4-I-M6-K-M8-L-X	45
S-A-M1-G-M7-J-X	43
S-B-M3-F-M4-I-M6-K-M8-L-X	52
S-B-M2-E-M7-J-X	40
S-D-M2-E-M7-J-X	35
S-D-M5-H-X	35

## 2.1.2 간트차트 (1/2)

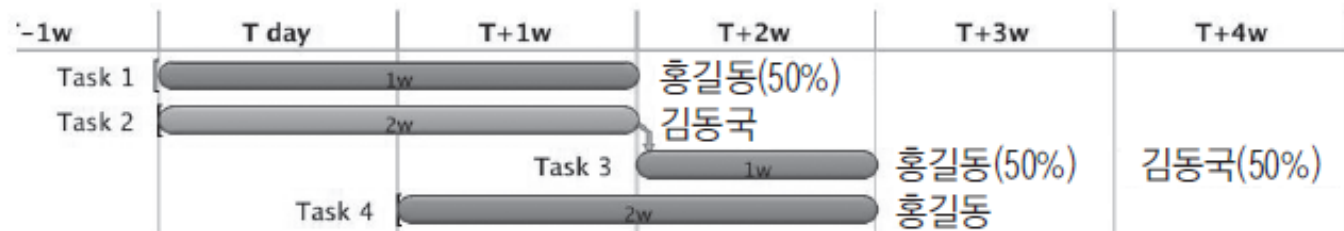
### ● 칸트 차트(Gantt Chart)

- 프로젝트의 스케줄링, 예산 산정, 자원 계획을 수립하기 위해 사용하는 일정 표현 기법
- 바(Bar) 차트 형태로 표시하며 바의 길이에 비례하여 소요 시간을 나타냄



## 2.1.2 간트차트 (2/2)

- 예비시간이나 계획 대비 진척도 관리에도 사용
- 소요자원의 할당을 통해 자원관리에도 활용



	T day	T+1w	T+2w	T+3w	T+4w
홍길동	20	20		T1	
			20	T3	
		40	40	T4	
홍길동 총합	20	60	60		
김동국	40	40		T2	
			20	T3	
김동국 총합	40	40	20		

## 2.1.3 애자일 프로세스의 스케줄링

- 애자일 프로세스는 **스토리 카드**를 주로 사용
  - CPM 네트워크와 간트차트는 전통적인 SW개발 프로세스에서 사용
  - 스토리카드의 내용 : **사용자스토리**, 비즈니스 **우선순위**, 스토리 점수, 연관된 스토리 등

반복#1 : 3주 4/1 ~ 4/21	반복#2 : 3주 4/22 ~ 5/12	반복#3 : 3주 5/13 ~ 6/2	반복#4 : 3주 6/3 ~ 6/23
UC1 : 스토리1 점수 : 5 우선순위 : 1 선행 : 없음	UC3 : 스토리3 점수 : 2 우선순위 : 1 선행 : UC9, UC1	UC2 : 스토리2 점수 : 4 우선순위 : 4 <u>선행 : UC8</u>	UC5 : 스토리5 점수 : 3 우선순위 : 4 선행 : UC4
UC9 : 스토리9 점수 : 4 우선순위 : 2 선행 : 없음	UC4 : 스토리4 점수 : 5 우선순위 : 2 선행 : UC3	<b>선후관계</b> UC8 : 스토리8 점수 : 5 우선순위 : 3 선행 : 없음	UC7 : 스토리7 점수 : 5 우선순위 : 4 선행 : UC2

## 2.2 비용 관리 기법

- 소프트웨어 개발 비용의 정확한 예측이 어려움
- 노력(Effort) : 작업이 완료되기 위해 필요한 노동의 양
  - 노력의 단위 : MM(man-month)
  - 프로젝트에 투입되는 월-인원을 나타냄
  - 프로젝트의 크기를 표현할 때 주로 사용

ex) 5MM = 한사람이 일한다면 5달이요



뭔가 아래에 설명이 더 있었는데...  
거꾸로해보기

## 2.2.1 COCOMO (1/5)

비용추정모델

- COCOMO/COCOMO II(Constructive Cost Model)
  - 규모(LOC)를 기반으로 하는 수학적 공식을 사용
  - 여기에, 개발 소프트웨어의 유형이나 소프트웨어에 영향을 주는 요인을 고려하여 보정

$$\text{노력}(E) = A \times (\text{Size})^B \times M$$

- A: 소프트웨어 유형에 좌우되는 상수
- size: 개발될 소프트웨어의 원시코드의 라인수(KLOC 사용)   
 *→ 프로그래밍 언어*
- B: 1에서 1.5 사이의 값
- M: 보정계수

## \* COCOMO 단점

- LOC 산정 어려움, 돌리면 결과가 틀림
- 프로젝트유형구분어려움

## 2.2.1 COCOMO (2/5)

개발유형에 따라 가지

### ● COCOMO 표준 산정 공식

프로젝트 유형	노력	기간
유기형	$PM = 2.4 \times (KDSI)^{1.05}$	$TDEV = 2.5 \times (PM)^{0.38}$
반결합형	$PM = 3.0 \times (KDSI)^{1.12}$	$TDEV = 2.5 \times (PM)^{0.35}$
임베디드형	$PM = 3.6 \times (KDSI)^{1.20}$	$TDEV = 2.5 \times (PM)^{0.32}$

- **KDSI**(Kilo Delivered Source Instructions) = KLOC
- PM(Person-Month) = MM ← 계산하려면 KLOC를 알아야함. 근데 개발 전에 어떻게?
- TDEV : 프로젝트를 종료하는 데 소요되는 개월 수  
→ 비슷한 기준 계통을 활용

## 2.2.1 COCOMO (3/5)

- 보정(scaling) 계수 M

- 프로젝트에 영향을 주는 다른 요소들을 고려하여 비용 승수(cost-driver)를 결정
- 보정 예: 프로그램의 경험이 많다면 노력 예측 값이 작아져야 하고, 실시간 처리 요구가 있다면 예측 값이 커져야 함
- 각 승수의 값을 예측된 PM 값에 곱하여 계산
- 비용 승수 : 일반적으로 0.9~1.5 사이 값



## 2.2.1 COCOMO (4/5)

비용승수 요소		승수값					
		매우 낮음	낮음	보통	높음	매우 높음	극히 높음
제품 특성	요구되는 신뢰도	0.75	0.88	1	1.15	1.4	
	데이터베이스 크기		0.94	1	1.08	1.16	
	제품의 복잡도	0.7	0.85	1	1.15	1.3	1.65
컴퓨터 특성	실행시간 제약			1	1.11	1.3	1.66
	주기억장치 제약			1	1.06	1.21	1.56
	HW, SW 안전성		0.87	1	1.15	1.3	
	처리시간		0.87	1	1.07	1.15	
개인 특성	분석가의 능력	1.46	1.19	1	0.86	0.71	
	응용경험	1.29	1.13	1	0.91	0.82	
	컴퓨터와의 친숙성	1.21	1.1	1	0.9		
	프로그래머 능력	1.42	1.17	1	0.86	0.7	
	프로그래밍언어 경험	1.14	1.07	1	0.95		
프로젝트 특성	SE 원리의 사용	1.24	1.1	1	0.91	0.81	
	SW 도구의 사용	1.24	1.1	1	0.91	0.83	
	요구되는 개발일정	1.23	1.08	1	1.04	1.1	

초기이탈로 인해  
 → COCOMO도 등장  
 • KLOC로 PM을 계산  
 → KLOC 추정을 잘못하면 문제  
 → LOC는 사용하지 않는  
 function point 사용

## 2.2.1 COCOMO (5/5)

- 예 : 33,360 LOC로 예상되는 반결합형 프로젝트
  - 개발노력(PM) =  $3.0 \times (\text{KDSI})^{1.12} = 3.0 \times (33.36)^{1.12}$   
 $= 152.45 = 153 \text{ MM}(\text{인원/월})$
  - 보정: 비용 승수 요소 결정에 따라 scaling
    - $152.45 \times (1.14 \times 0.88) = 152.94$
  - 총개발시간(TDEV) =  $2.5 \times (\text{PM})^{0.35}$   
 $= 2.5 \times (152.94)^{0.35} = 14.5 \text{ M(월)}$
  - 소요인원(N) =  $152.94 / 14.5 = 10.5 \approx 11 \text{ 명}$
  - 생산성 = MM 당 생산하는 LOC =  $\text{LOC} / \text{MM}$   
 $= 33,360 / 152.94 \approx 218 \text{ (한 달에 218라인생산)}$

## 2.2.2 델파이 기법

- 델파이 기법
  - 전문가들의 의견수립, 중재, 타협의 방식으로 반복적인 피드백을 통한 하향식 의견 도출 방법으로 문제를 해결하는 기법
  - 1964년 미국의 RAND연구소에서 개발되어 IT분야, 연구개발분야, 교육분야, 군사분야 등에서 활용
- 노력 추정을 위한 델파이 기법 적용
  - 다수의 전문가가 모여 소프트웨어의 요구사항을 기준으로 소프트웨어 크기를 예상

누가 했는지의 배고 동향 분석 전문가에게  
→ 회의

## 2.2.3 기능점수 산정법

- 코드가 작성되지 않은 단계에서 정확한 LOC의 예측은 불가능  $\Rightarrow$  LOC로 생산성이나 품질을 평가하기는 곤란함
- 기능 점수(FP, function points) *기능에 정량화*
  - 구현되는 언어에 관계없는 메트릭(Metric)
  - 소프트웨어 시스템이 제공하는 기능의 필요 정도와 복잡도를 기준으로 평가하는 방식
  - 국내에서는 2012년 이후 매년 SW산업협회에서 SW사업 대가 산정 가이드 제정 공표  $\Rightarrow$  IFPUG에서 정의한 FP모델을 기준(3장에서 자세하게 다룸)

## 2.3 위험 관리 (1/4)

- 위험(risk)이란?
  - 요구사항의 빈번한 변경, 프로젝트 팀원의 부적절성과 같이 재작업, 의사소통의 ~~증가~~<sup>복재</sup>와 같은 문제를 유발하는 요인
  - 프로젝트 성과를 저하시키는 요인
- 리스크를 사전에 식별하고 관리해야 함

프로젝트 위험 처리를 위한 양식 (예시)

No	리스크	치명도	발생 빈도	완화 계획	대안	담당자
1	팀원 이탈(이직)	High	Mid	작업 부하 경감	다중 역할 부여	잡스
2	빈번한 요구사항 변경	Mid	High	프로토타이핑	베이스라인 통제	머스크
3	낮은 기술 수준	Mid	Mid	직무 매칭	교육 훈련	세르게이

## 2.3 위험 관리 (2/4)

- Boehm이 정의한 10대 위험요소
  - 인력 부족(personnel shortfalls)
  - 비현실적 일정 및 예산
  - 잘못된 기능과 특징 개발
  - 잘못된 인터페이스 개발
  - 과포장(필요하지 않은 좋은 기능 추가)
  - 계속적인 요구변경
  - 외부에 보일만한 컴포넌트 빈약
  - 외부에서 관찰 할 만한 기능의 빈약
  - 실시간 성능의 빈약
  - 낡은 기술

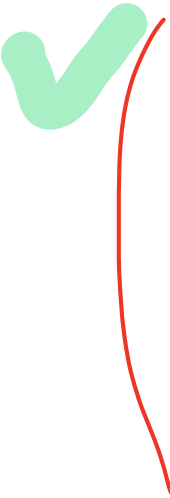
## 2.3 위험 관리 (3/4)

- 위험 평가 : 각 위험에 대한 피해 정도, 위험 해결 방법, 이에 대한 비용들을 결정하는 것
  - 손실 발생 확률
  - 손실 발생 규모
  - 위험 노출(exposure)
- 영향도에 따라 평가하고 우선순위를 매김
  - 정량적 방법 : 리스크 확률을 고려한 영향을 비용으로 환산
  - 정성적 방법 : 주관적인 판단으로 평가 (발생확률에 대한 정보가 없을 때)

## 2.3 위험 관리 (4/4)

- 위험 요소를 해소하기 위한 방법을 강구하고 프로젝트 실행하는 동안 이를 적용

- 방법

- 
- 리스크를 피하기 위하여 계획을 변경
  - 책임을 다른 기관에 맡김
  - 프로토타이핑
  - 유능한 인재를 등용
  - 3자와 협업



### 3. 프로젝트 조직 (1/2)

- 조직의 구성
  - 소프트웨어 개발 생산성에 큰 영향
  - 작업의 특성과 팀 구성원 사이의 의사교류
- 프로젝트 팀 조직 정의
  - 역할과 책임이 어디에 있는가?
  - 어떤 통로로 정보가 전달되고 결정되는가?
- 프로젝트 구성원의 역할
  - 프로젝트 관리자(PM)
  - 프로젝트 팀장(TL)
  - 형상 관리자(CM)
  - 리어종 그룹(Liaison Group)
  - 프로젝트 리더(PL)
  - 프로그램 엔지니어(PE)
  - 품질 관리자(QE)

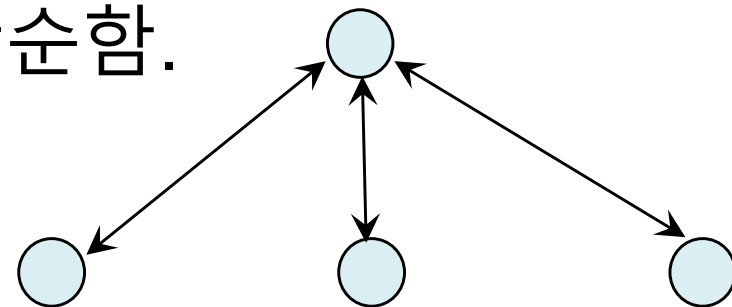
### 3. 프로젝트 조직 (2/2)

- 프로젝트 팀 구조의 유형



## 3.1 중앙집중형 구조

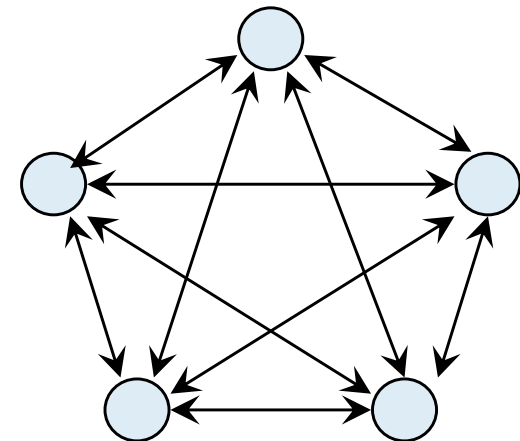
- 중앙집중형 팀 구조는 프로젝트에서 수행해야 할 작업 목록이 단순하거나 충분히 이해된 경우에 적합
- 의사결정권이 리더에게 집중
- 계층적 팀 구조
- 유능한 프로젝트 리더가 필요
- 문제 해결이 신속하게 이루어질 수 있고, 의사소통의 패턴이 매우 단순함.



의사 교환 경로

## 3.2 분산형 팀 구조

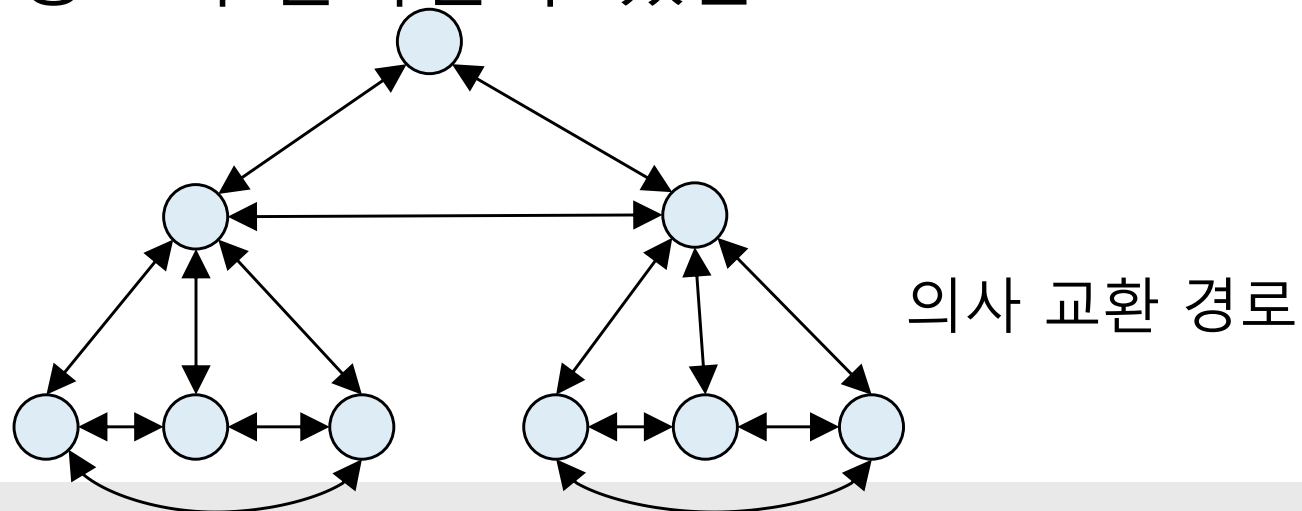
- 프로젝트의 주요 의사결정이 팀 구성원의 합의에 의해 이루어지는 **민주주의적 팀 구성**
  - 서로 협동하여 수행하는 Ego-less 팀
  - 자신이 일을 알아서 수행
- 의사소통 패턴이 복잡**하므로 대규모 구성원을 포함하는 프로젝트에는 적합하지 않음
- 장기 프로젝트에 적합



의사 교환 경로

## 3.3 하이브리드 팀 구조

- 중앙집중형 팀 구조와 분산형 팀 구조를 통합한 계층형 팀 구조
- 프로젝트 관리자 : 각 팀의 리더와 중요한 의사결정을 하기 위한 중앙집중형 구조
- 팀 내부의 운영 : 분산형 구조를 채택하여 의사소통
- 의사 전달 경로가 길어질 수 있음



## 3.4 전사적 운영 조직

- 품질 관리 팀

- 프로젝트 수행 과정에서 산출물에 대한 리뷰, 인스펙션, 테스트 등과 같은 활동 지원
- 프로젝트 수행 경험이 있고 관련 분야의 전문적인 지식을 확보한 중급 이상의 엔지니어들로 구성

- PMO(Project Management Office)

- 전사 차원에서 프로젝트 생성, 진행 과정의 모니터링, 프로젝트 단위 리스크 해결 지원 등과 같은 업무를 수행하는 조직

# 4. 프로젝트 관리 계획서

## 1. 개요(Introduction)

- 1.1 문서의 목적
- 1.2 프로젝트 개요
- 1.3 관련 문서, 용어 및 약어

## 2. 개발 계획(Development Plan)

- 2.1 프로세스 모델 및 개발 방법론
- 2.2 작업 분할도(Work Breakdown Structure)
- 2.3 소요 자원(Staffing, Cost)
- 2.4 일정 계획(Gantt Chart)

## 3. 조직 구성(Organization)

- 3.1 팀 구조
- 3.2 책임 및 역할

## 4. 기술 관리 계획(Technical Management)

- 4.1 버전 관리
- 4.2 형상 관리
- 4.3 기술 관리

## 5. 품질 관리 계획(Quality control)

- 5.1 품질 목표
- 5.2 검토 방법 및 주기
- 5.3 품질 관리 적용 기법
- 5.4 리스크 관리

## 6. 개발 환경(Development Environment)

- 6.1 요구되는 소프트웨어 스펙
- 6.2 요구되는 하드웨어 스펙
- 6.3 개발 공간 및 보안 요구사항

## 7. 개발 산출물(Deliverables)