Chapter 3 .

소프트웨어 개발 계획

Software Planning : 소프트웨어 개발에 드는 기간과 비용을 산정하는 방법

신한대학교
2015 봄학기
소프트웨어 공학
고덕윤
maniara.k@gmail.com

0 소프트웨어 프로젝트 관리

- 소프트웨어 개발 프로젝트를
  - 조직하고(organizing)
  - 계획하고(planning)
  - 일정관리(scheduling) 을 수행하는 행위

- 계획 수립
  - 일정, 비용, 조직 구성, 생산품 등을 사전에 계획하는 것
  - 제품 생산의 작은 부분을 정의하고 순서를 결정함
  - 일정/비용 예측 및 리스크 분석
  - 되도록 현실적/구체적 계획을 세운다.
  - 비용/기술적인 측면을 고려해야 한다.

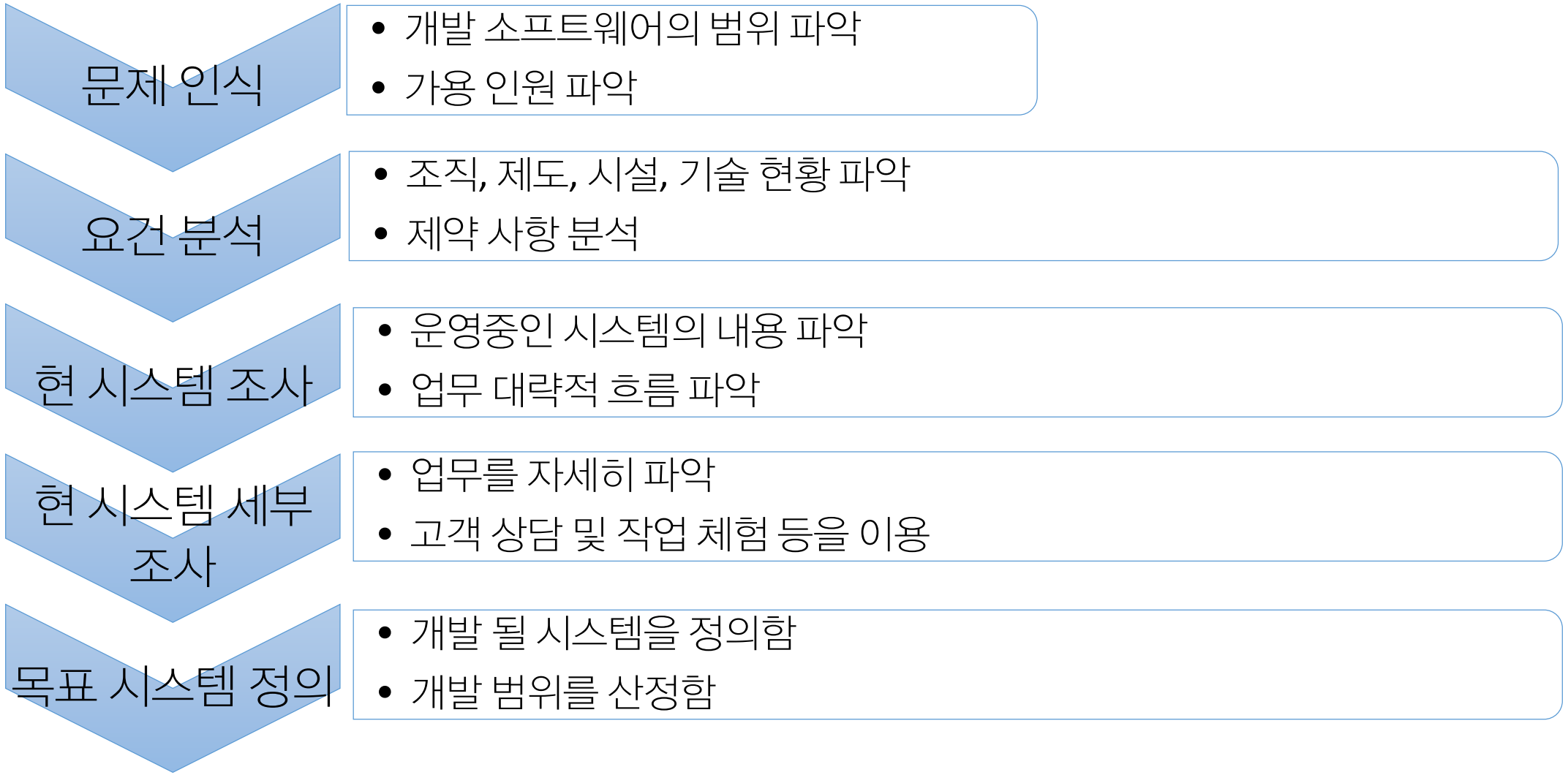
1 문제 정의 (Project Scoping

1.1 프로젝트 스코핑 : 개발 범위를 정하는 일

- 사용자의 입장에서 개발 될 소프트웨어를 미리 고민함.
  - 알고리즘, 데이터베이스 모델, 하드웨어 등의 전문 용어는 피한다.
- 기능, 성능, 제약조건, 인터페이스 등의 근거가 됨.
- 사용자와의 면담이나 현장 관찰 등을 수행.

- 문제 정의서 : 개발의 문제점 및 현재 상황 등을 기술한 문서

1.2 스코핑 프로세스


문제 인식
• 개발 소프트웨어의 범위 파악
• 가용 인원 파악
요건 분석
• 조직, 제도, 시설, 기술 현황 파악
• 제약 사항 분석
현 시스템 조사
• 운영중인 시스템의 내용 파악
• 업무 대략적 흐름 파악
현 시스템 세부 조사
• 업무를 자세히 파악
• 고객 상담 및 작업 체험 등을 이용
목표 시스템 정의
• 개발 될 시스템을 정의함
• 개발 범위를 산정함

1.3 목적 수립 및 시스템 정의

- 신규 시스템의 목표 설정
  - 기대효과
  - 투자 효과

- 시스템 정의
  - 문제 기술
  - 시스템의 필요성 기술
  - 시스템의 목표
  - 제약사항
  - 제공 기능
  - 개발, 유지보수 환경 정의

1.4 타당성 분석

- 경제적 타당성
  - 투자 효율성
  - 시장성
  - 비용대비 수익의 비교
- 기술적 타당성
  - (실패) 사례 연구
  - 모의 실험
  - 프로토타입 제작
- 법적 타당성
  - 시장, 관행들 조사
  - 라이선스 조사

2 비용 산정

2.1 소프트웨어 개발의 비용 예측

- 정확한 비용 예측은 불가능함
  - 벽돌, 알루미늄 등의 원자재가가 없으므로, 상대적으로 인건비 비율이 높음
  - 예측 불가능한 요소가 많음
  - 사례가 많지 않음
  - 경험에 의존도가 큼

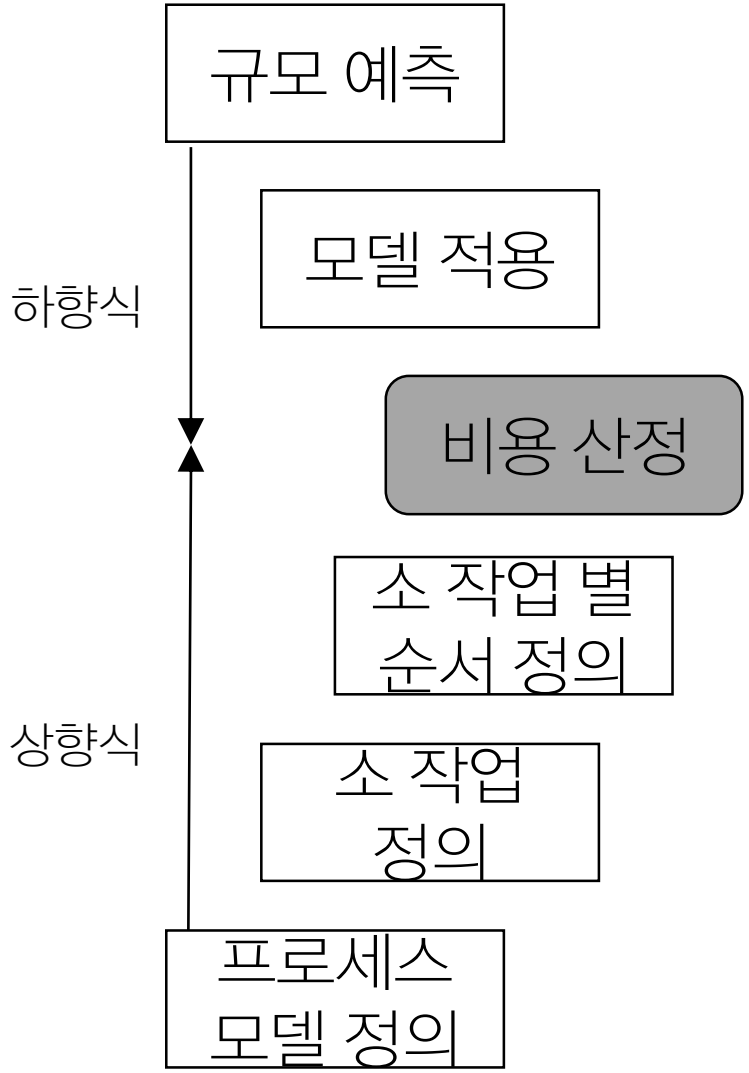
- 소프트웨어 개발에 필요한 예산 항목
  - 인건비 : 월별 투입 인원수에 기초함
  - 경비 : 여행비, 인쇄비, 재료비 등
  - 간접비 : 예측 하지 못한 상황에 발생하는 비용

2.2 비용 측정 요소

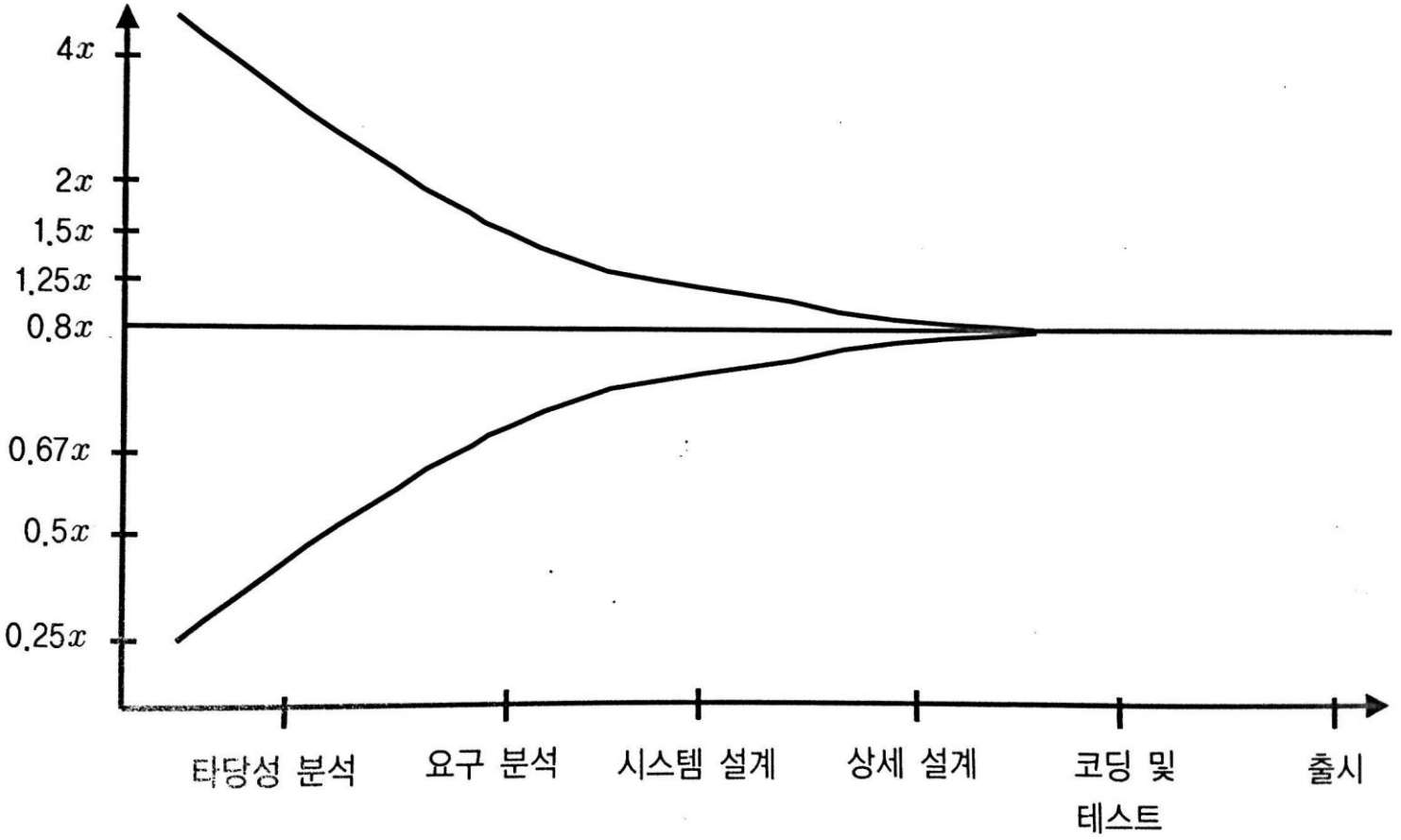
- 제품의 크기
  - 크기가 커지면 비용이 기하급수적으로 늘어남
- 제품의 복잡도
  - 응용 : 개발 지원 : 시스템 = 1:3:9
- 프로그래머의 자질
  - 코딩, 디버깅 능력
  - 언어와 응용능력의 친숙도
- 요구되는 품질 수준
  - 실시간/고성능/안전 소프트웨어
- 개발 시간
  - “프로젝트의 비용은 개발 기간의 4제곱의 반비례 "

2.3 프로젝트 비용 예측 방법

- 상향식
  - 소요 기간을 구하고 투입할 인원을 통해 최종 비용 계산
  - 소 작업(activity) 비용을 일일이 예측함
  - 비용 산정자의 주관적 요소가 크게 작용함
- 하향식
  - 프로그램 규모를 통해 규모를 예측하고 인력과 기간을 추정
  - 코드 수나 기능 수를 기준으로 예측함



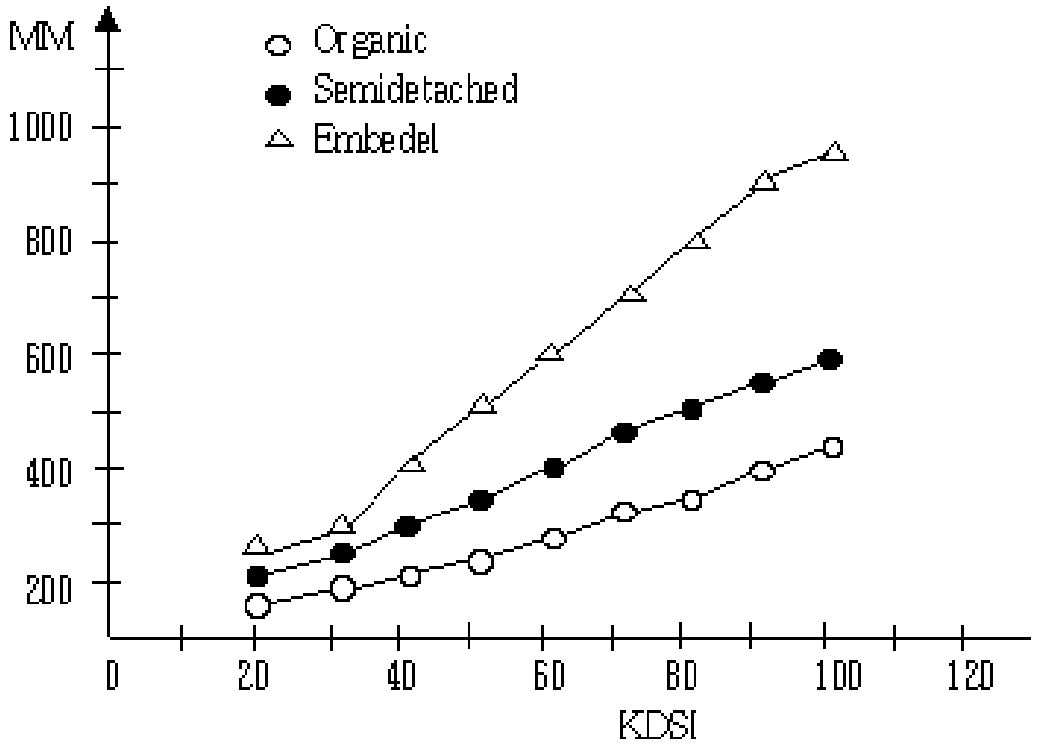
2.4 비용 추정의 불확실성



- 프로젝트 후반으로 갈 수록 정확한 예측이 가능해 짐
- 가장 초기에는 최대 4배의 비용 차이가 발생할 수 있음
- 비용 추정은 불확실할 수 밖에 없음
- 실제 비용과 예측 비용의 차이는 시간이 흐를수록 적어짐

2.5 COCOMO(COnstructive COst MOdel) model

- COCOMO I(1981), COCOMO II(1995) 발표
- B. Boehm 이 TRW의 광범위한 자료를 통해 분석
- 프로그램 규모(KDSI:Kilo Delivered Source Instruction) 기반
- 프로젝트 유형 기반(mode)
  - 단순형(Organic) : 중소규모, 자료처리, 계산용, 비즈니스 자료 등
  - 중간형(Semi-detached) : 운영 체제, DBMS 등
  - 임베디드형(Embedded) : 하드웨어가 포함, 미사일 유도 시스템, 도시가스 제어 등



2.6 COCOMO : 프로젝트 유형별 공식

Mode	노력(MM)	기간(D)
Organic	$PM = 2.4 \times (KSDI)^{1.05}$	$\mathrm{TDEV} = 2.5 \times (\mathrm{PM})^{0.38}$
Semi–detached	$PM = 3.0 \times (KSDI)^{1.12}$	$\mathrm{TDEV} = 2.5 \times (\mathrm{PM})^{0.35}$
Embedded	$PM = 3.6 \times (KSDI)^{1.20}$	$\mathrm{TDEV} = 2.5 \times (\mathrm{PM})^{0.32}$

- 예제 : CAD 시스템 예상 규모 : 33,360 LOC
  - $PM = 3.0 \times (KSDI)^{1.12} = 3.0 \times 33.3^{1.12} = 152\ PM(MM)$
  - $TDEV = 2.5 \times (PMI)^{0.35} = 2.5 \times 153^{0.35} = 14.5\ TDEV(M)$

- 연습문제 : 응용프로그램 예상규모 : 50,000 LOC

2.7 COCOMO의 세가지 모델

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

2.8 Intermediate COCOMO : Cost Driver

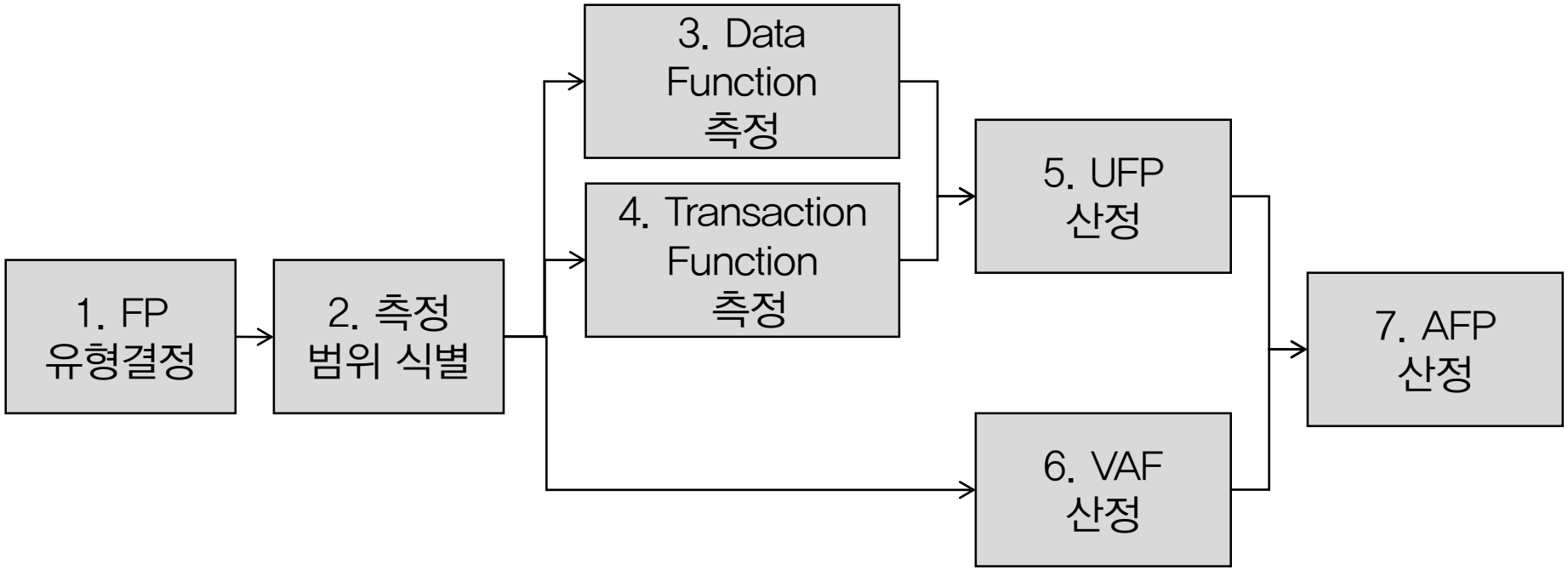
- 제품 특성 요소
  - 요구되는 신뢰도
  - 데이터베이스 크기
  - 제품 복잡도
- 컴퓨터의 특성
  - 실행시간 제약
  - 주기억 장치 제약
  - 안전성
  - 처리시간
- 개발자 특성
  - 분석가의 능력
  - 응용 경험
  - 컴퓨터와의 친숙성
  - 프로그래밍 언어 경험
- 프로젝트 성격
  - 소프트웨어 공학 원리 사용
  - 소프트웨어 도구 사용
  - 요구되는 개발 일정

Cost Driver	비율					
	매우낮음	낮음	보통	높음	매우높음	극히매우높음
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
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	
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		
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	

2.9 기능 점수 (Function Points)

- 1979년 Allen J. Albercht 가 제안
- 물리적 규모 측정의 한계를 극복하고자..
  - 정확한 라인 수 산정 불가능
- 사용자 관점에서의 SW 규모 측정 방법
  - 고객은 라인 수에 관심이 없음
  - 고객에 입장에서 납득하는데 어려움이 있음
- 다양한 적용 범위
  (벤치마킹 조사, 개발비용 산정, 품질측정 등)
- CFPS 자격 인증이 있음

2.10 기능 점수 작성 순서



용어
–FP : Function Point
–UFP : Unadjusted Function Point
–VAF : Value Adjustment Factor
–AFP : Adjusted Function Point

2.11 기능 점수 계산 사례

- 별도 자료 참조

3 일정 계획

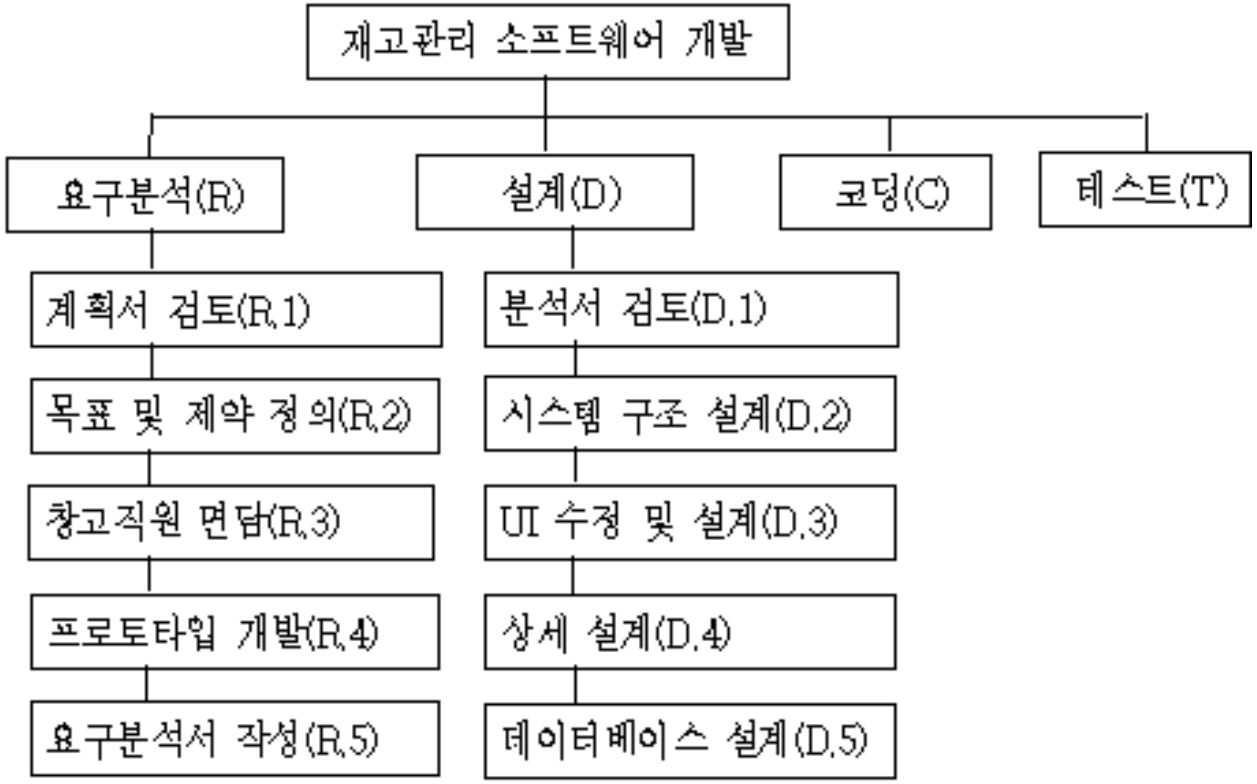
3.1

- 소 작업(activity)을 파악하고 일정을 정함

- 일정계획(Scheduling) 프로세스
  - 작업 분해(Work Breakdown Structure)
  - CPM(Critical path method) 네트워크 작성
  - 각 작업의 소요 기간을 산정함
  - 간트차트 (Gantt–chart) 작성

3.2 작업 분해(Decomposition)

- 프로젝트에 필요한 소 작업(activity)를 식별함
- WBS(Work Break Structure) : 계층적 작업 목록 작성

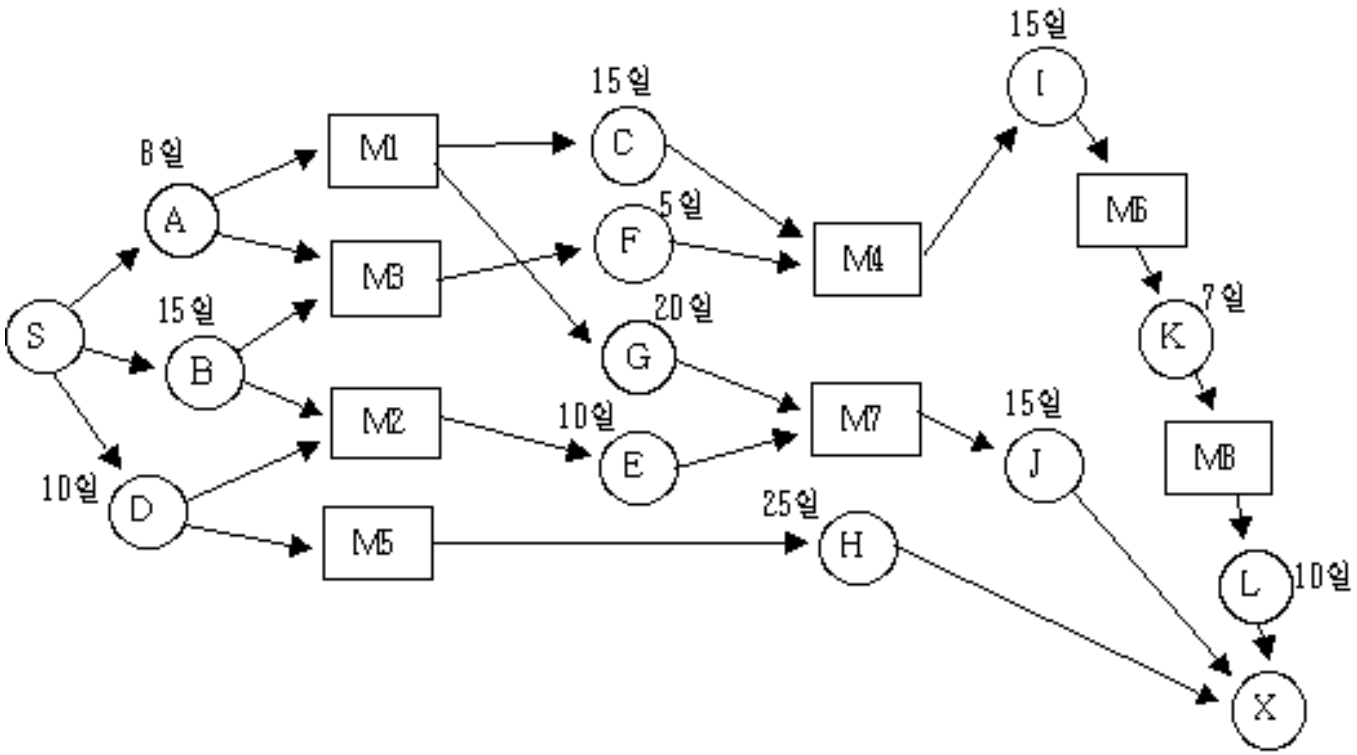


3.3 CPM 네트워크

- CPM(Critical Path Method)는 소 작업(activity)의 순서를 나타냄
- 앞의 작업이 끝나지 않으면 뒤의 작업을 시작할 수 없다.
- 일정표 작성에 필수적인 작업
- CPM 네트워크 프로세스
  - 모든 소 작업과 선행작업, 소요기간을 정한다.
  - 소 작업의 선후관계를 CPM 네트워크를 작성한다.
  - 최적 경로를 계산한다.

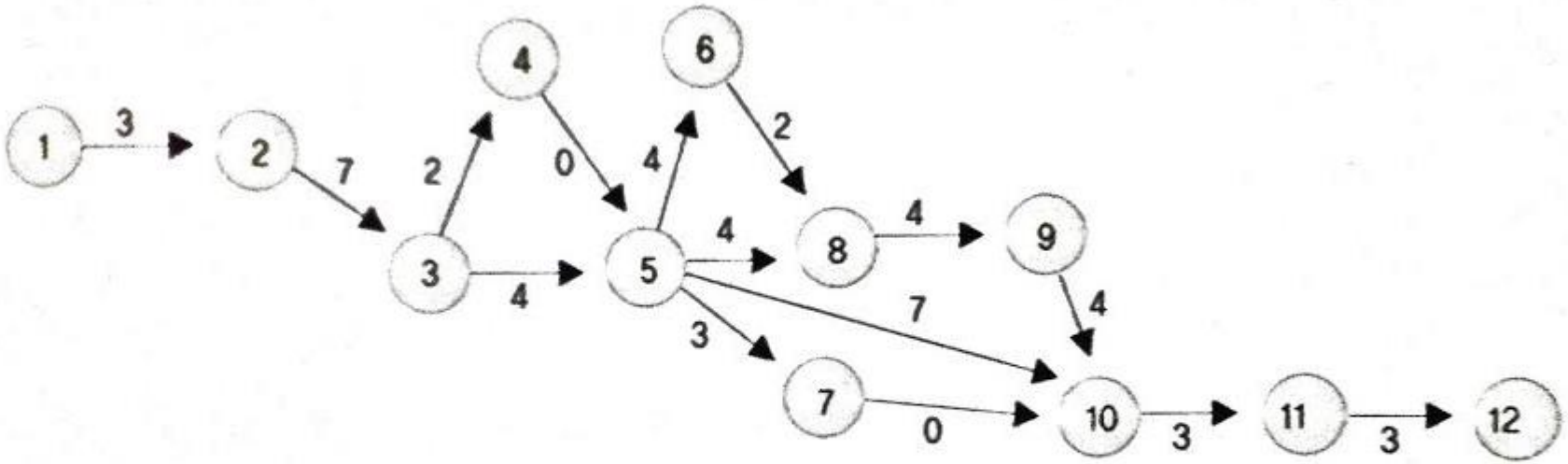
소작업	선행작업	소요기간(일)
A	-	8
B	-	15
C	A	15
D	-	10
E	B, D	10
F	A, B	5
G	A	20
H	D	25
I	C, F	15
J	G, E	15
K	I	7
L	K	10

3.4 CPM 예제



가능 경로	소요 기간(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

** 연습 문제



(1) Critical path를 찾아보시오.

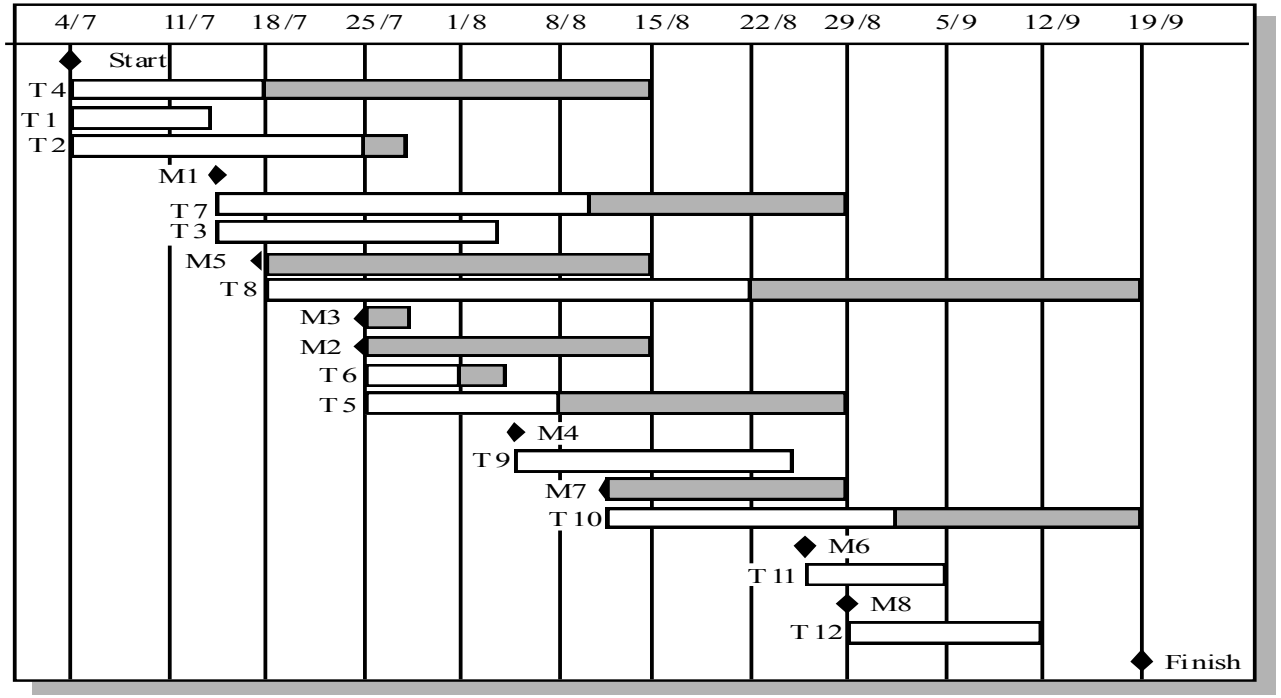
(2) 정상적인 예정 기간(normal time estimate)은?

3.5 CPM의 장점

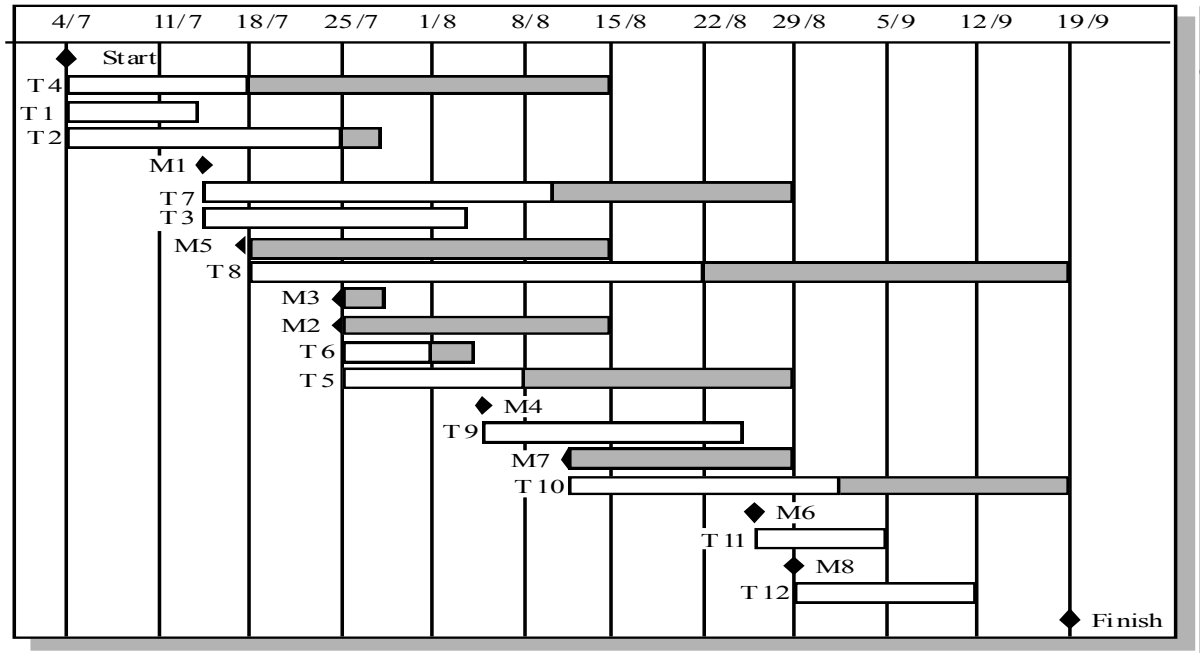
- 관리자의 일정 계획에 도움을 준다.
- 프로젝트의 작업 사이의 관계를 나타내고 최단 경로를 파악할 수 있게 해준다.
- 병행 작업 식별이 가능하다.
- 프로젝트 일정 관리를 쉽게 해 준다.

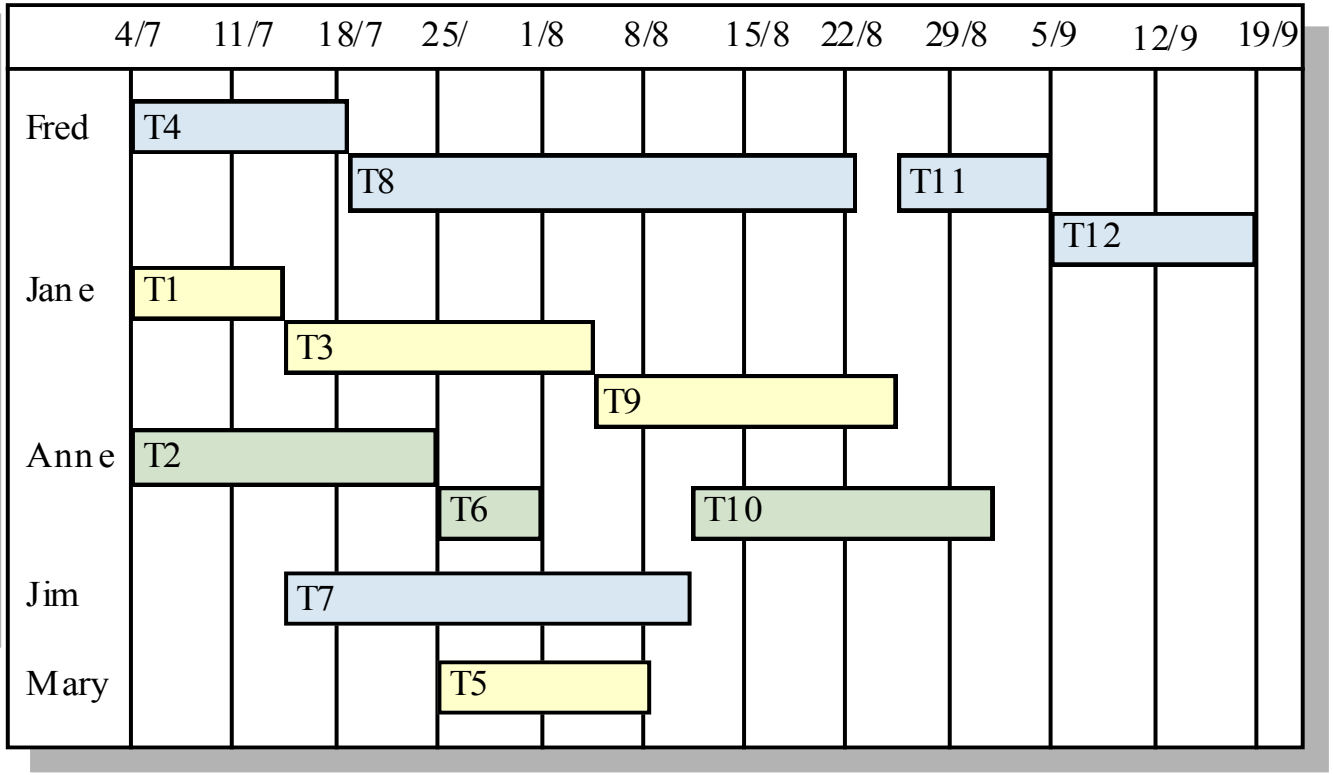
3.6 간트 차트(Gantt chart)

- 작업과 시간 일정표를 실제 날짜에 맞게 도식화 한 차트.
- CPM의 결과에 따라 각 소 작업의 시작일, 기간 등을 작성한다.
- 예비 시간과 임계 시간을 모두 표시한다.
  - 예비시간 : 시작 가능한 날짜를 나타낸다.
  - 임계시간 : 해당 시점에는 꼭 작업을 시작해야 한다.



3.7 간트 차트 작성의 예


4/7
11/7
18/7
25/7
1/8
8/8
15/8
22/8
29/8
5/9
12/9
19/9
Start
T4
T1
T2
M1
T7
T3
M5
T8
M3
M2
T6
T5
M4
T9
M7
T10
M6
T11
M8
T12
Finish


4/7
11/7
18/7
25/
1/8
8/8
15/8
22/8
29/8
5/9
12/9
19/9
Fred
T4
T8
T11
T12
Jane
T1
T3
T9
Anne
T2
T6
T10
Jim
T7
Mary
T5

4 조직 계획

4.1

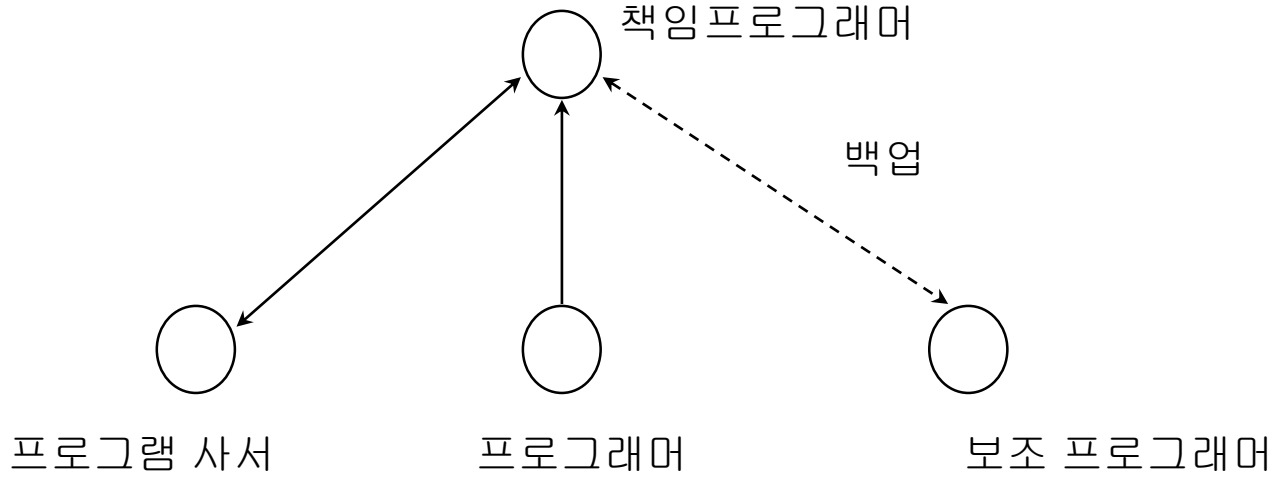
- 조직의 구성
  - 다른 제품과 달리 소프트웨어 생산성은 조직 구성의 영향을 크게 받음
  - 개발 기간과 밀접한 관계를 갖는다.
  - 개발자의 경험분포도 고루 분포 되어야 한다.
  - 3~8명의 팀이 가장 이상적이다.
  - 8명이 초과할 경우 중간 관리자를 두는 것이 좋다.

4.2 책임 프로그래머 팀 구성

- 외과 수술 팀 구성에서 따옴
- 중앙 집중형 구조이다.
- 관리자의 지시에 따라 작업을 수행
- 프로그래머는 관리자에게 보고함
- 소규모 문제에 적합함
- 역할
  - 책임 프로그래머(Chief programmer) : 제품설계, 주요부분 코딩, 중요 기술적 결정, 작업 지시
  - 프로그램 사서(Program librarian) : 프로그램 리스트 관리, 설계 문서 및 테스트 계획 관리
  - 보조 프로그래머 : 기술적 문제에 대해 상의, 고객/품질보증팀 등 다른 그룹과 접촉, 부분적 구현을 담당
  - 프로그래머 : 각 모듈을 구현

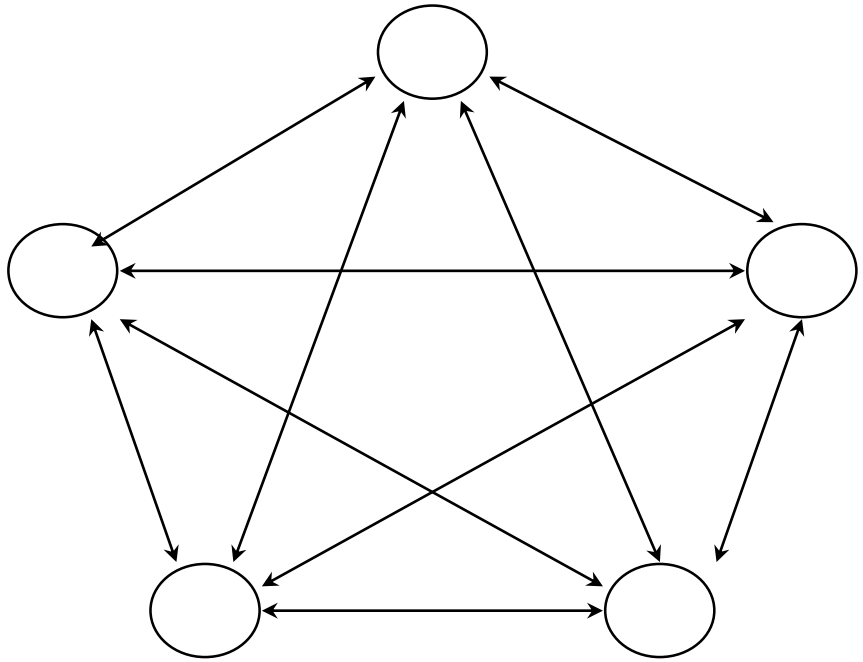
4.3 책임 프로그래머 팀 구성의 특징

- 장점
  - 의사 결정이 빠름
  - 소규모 프로젝트에 적합
  - 초보 프로그래머를 훈련시키는 기회로 적합
- 단점
  - 한 사람의 능력과 경험이 프로젝트의 성패 좌우
  - 보조 프로그래머의 역할이 모호



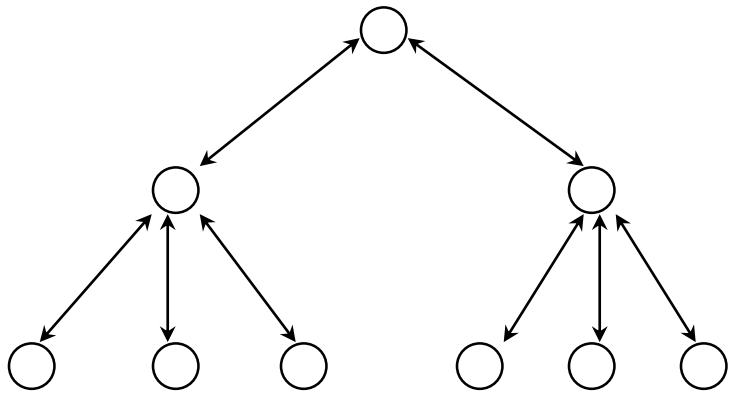
4.3 분산형 팀 구성

- 민주주의 식 의사결정
  - 서로 협동하여 수행하는 비이기적인 팀(Ego-less)
  - 자신이 있는 일을 알아서 수행
  - 구성원이 동등한 책임과 권한
- 의사 교환 경로
- 특징
  - 작업 만족도 높음
  - 의사 교류 활성화
  - 장기 프로젝트에 적합
- 단점
  - 책임이 명확하지 않은 일이 발생
  - 대규모에 적합하지 않음(의사 결정 지연 가능)



4.4 혼합형 팀 구성

- 집중형, 분산형의 단점을 보완
- 특징
  - 초보자와 경험자를 분리
  - 프로젝트 관리자와 고급 프로그래머에게 지휘권한이 주어짐
  - 의사교환은 초보 엔지니어나 중간 관리층으로 분산
- 소프트웨어 기능에 따라 계층적으로 분산
- 단점
  - 기술인력이 관리를 담당
  - 의사 전달 경로가 김



5 위험 분석(Risk Analysis)

5.1 10 가지 손실 요소

Top 10 Software Risk Items

1. Personnel shortfall
2. Unrealistic schedules and budgets
3. Developing the wrong functions and properties
4. Developing the wrong user interface
5. Gold-plating
6. Continuing stream of requirements changes
7. Shortfalls in externally furnished components
8. Shortfalls in externally performed tasks
9. Real-time performance shortfalls
10. Straining computer-science capabilities

5.2 10가지 손실 관리 포인트

Ten "suggestions"

1. Re–think the deadline given – is it for real?
2. Re–think the solution – is it incompatible with the deadline?
3. What is the requestor's real need/point of view?
4. Don't accept "expert" opinions blindly.
5. Determine which components really must be delivered at the deadline.
6. Don't cave–in, even as a last resort.
7. Change the solution to meet the deadline.
8. Make maximum use of existing systems and known technology.
9. Break the projects into earlier and smaller deliverables.
10. Don't forget to take credit for the success.

5.3 손실 계산

- 발생확률 * 손실비용(기간)

- 예) 4명의 팀원 중 1명이 입대할 확률 : 25%
  팀원이 입대할 경우 발생하는 손실 MM : 2MM
  0.25 * 2 = 0.5
  2주 정도의 여유기간을 할당하는 것이 좋음.

6 계획서 작성

6.1 계획서 구성

1 개 요
- 1.1 프로젝트 개요
- 1.2 프로젝트의 산출물
- 1.3 정의, 약어

2 자원 및 일정 예측
- 2.1 자원
  - 가. 인력
  - 나. 비용
- 2.2 일정

3 조직 구성 및 인력 배치
- 3.1 조직 구성
- 3.2 직무 기술

4 WBS

5 기술관리 방법
- 5.1 변경 관리
- 5.2 위험 관리
- 5.3 비용 및 진도 관리
- 5.4 문제점 해결 방안

6 표준 및 개발 절차
- 6.1 개발 방법론

7 검토 회의
- 7.1 검토회 일정
- 7.2 검토회 진행 방법
- 7.3 검토회 후속 조치

7 질문과 답변