3.3 일정 계획(Scheduling)

- 일정 계획

  개발 프로세스를 이루는 소작업(activity)를 파악하고 순서와 일정을 정하는 작업

  - 개발 프로세스 모형 결정
  - 단계별 소작업, 산출물, 이정표(milestone) 설정

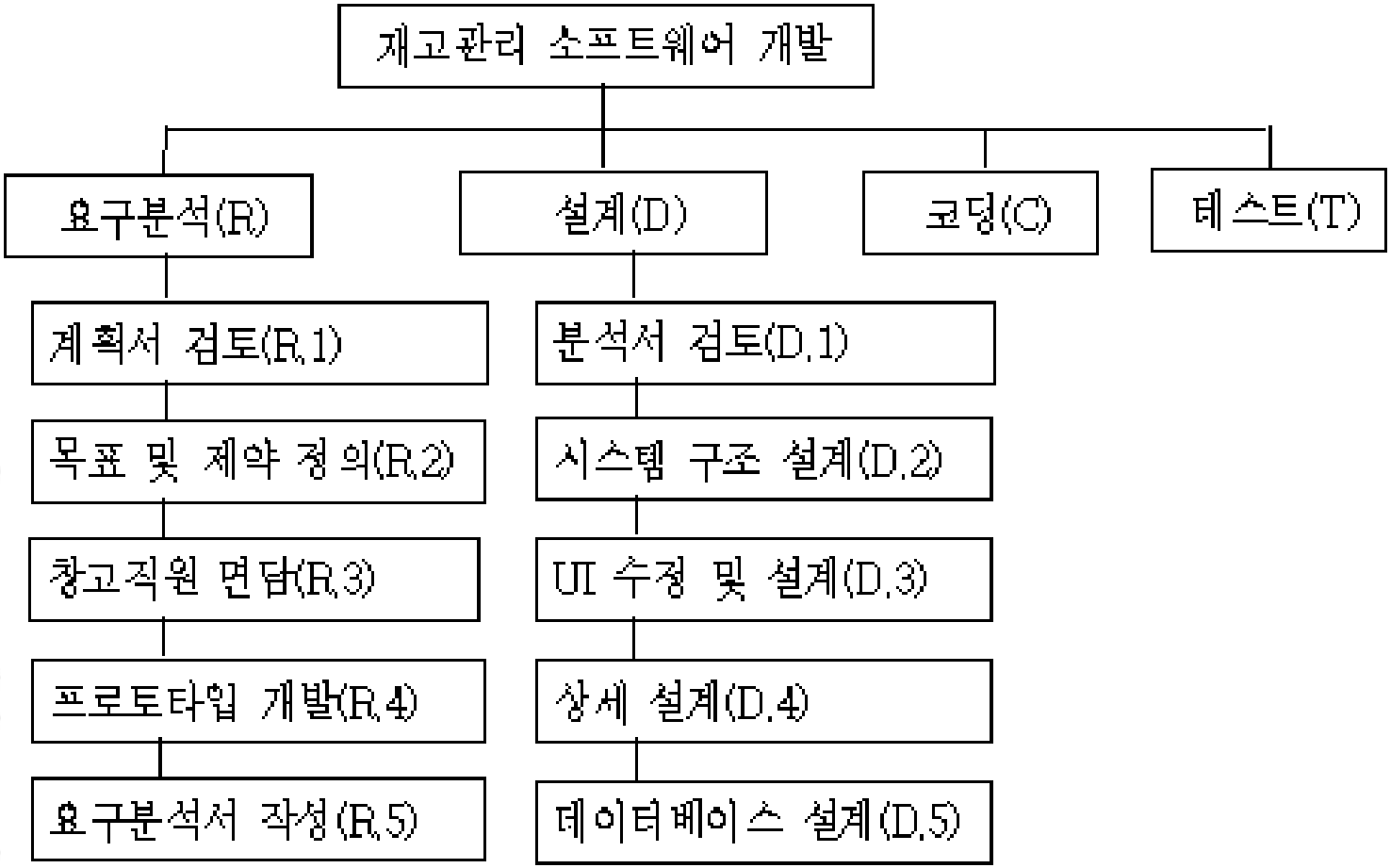
- 작업 순서
  - 작업분해(Work Breakdown Structure)
  - CPM 네트워크 작성
  - 최소 소요 기간을 구함
  - 소요 MM, 기간 산정하여 CPM 수정
  - 간트 차트로 그림

작업 분해(Decomposition)

- **작업 분해**

  프로젝트 완성에 필요한 activity를 찾아냄

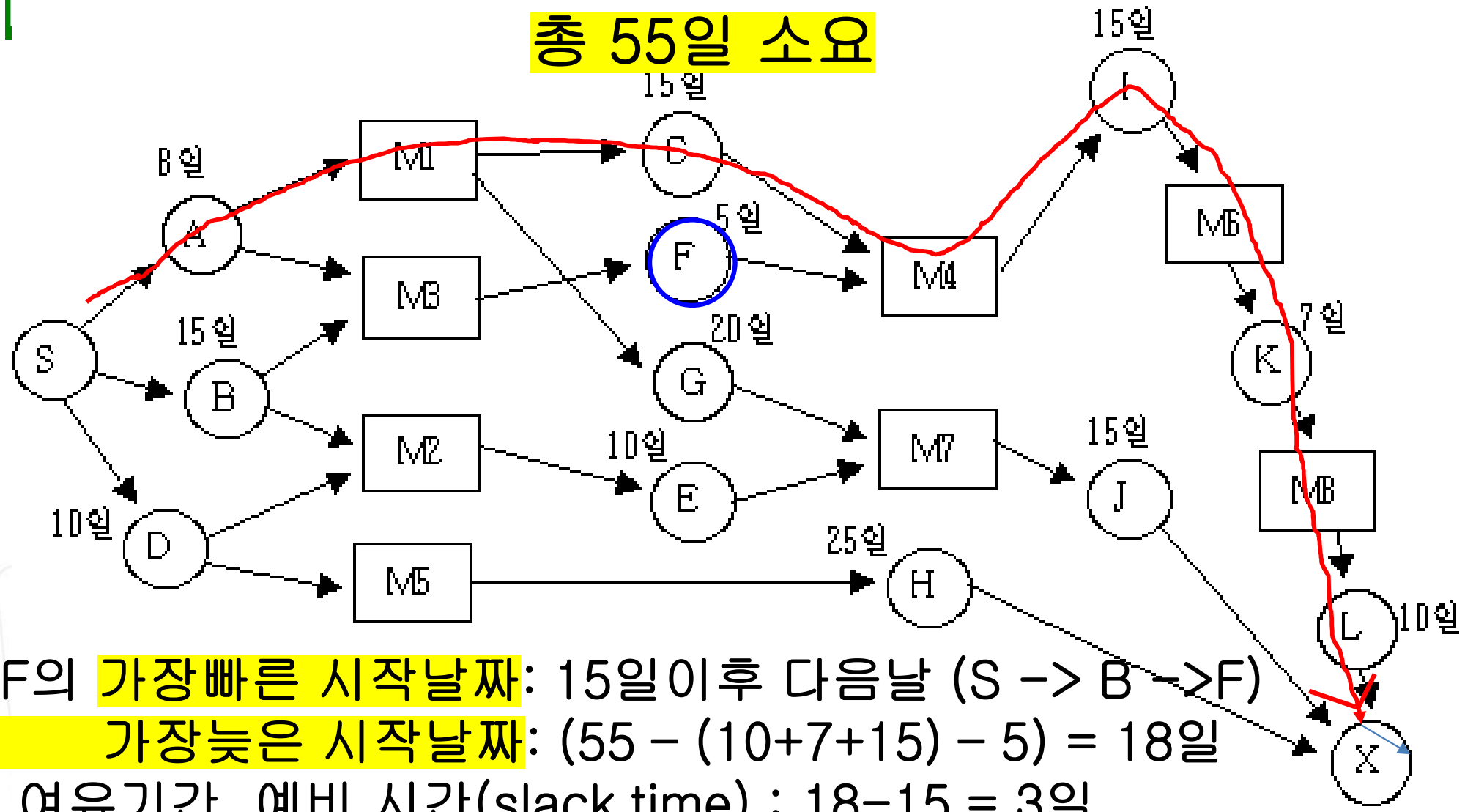
- **Work Breakdown Structure**
  - 계층적 구조



작업순서 결정 및 소요시간 예측

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

Activity 네트워크 : CPM (Critical Path Method)



F의 가장빠른 시작날짜: 15일이후 다음날 (S -> B ->F)

가장늦은 시작날짜: (55 – (10+7+15) – 5) = 18일

여유기간, 예비 시간(slack time) : 18-15 = 3일

C ? ➔ critical path 형성

임계 경로

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

How to find the longest path ?

Dijkstra's algorithm can be used if G has no negative weights. ➔ why ?

-Is it true ? G graph에 대해 shortest path를 구하게 되면, 그 path가 G graph의 longest path임.

CPM 네트워크

- 장점
  - 관리자의 일정 계획 수립에 도움
  - 프로젝트 안에 포함된 작업 사이의 관계
  - 병행 작업 계획
  - 일정 시뮬레이션
  - 일정 점검, 관리
- 관리에 대한 작업(예, 기자재구입, 인력 채용등) 도 포함 가능
- 작업 시간을 정확히 예측할 필요
- 소프트웨어 도구
  - MS-Project 등

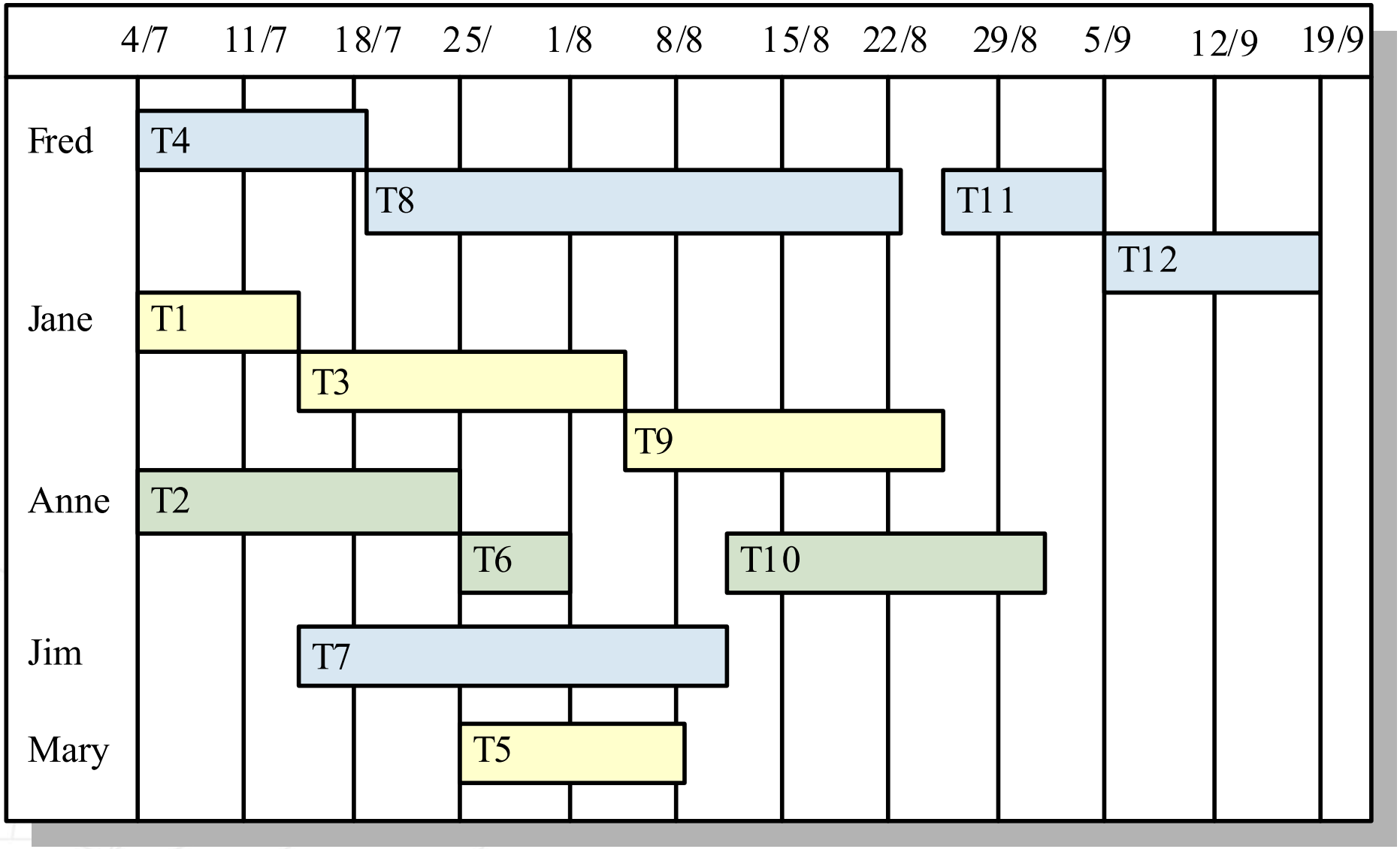
간트 차트

- 간트 차트
  - 소작업별로 작업의 시작과 끝을 나타낸 그래프
  - 예비시간을 보여줌
  - 계획 대비 진척도를 표시
  - 개인별 일정표

프로젝트 일정표


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
T1 1
M8
T12
Finish
예비 시간
(slack time)

Staff Allocation (일종의 간트 챠트)



애자일 계획

- 애자일 프로세스의 일정 계획 (스토리 카드 이용)

- 장점
  - 높은 우선순위를 가진 사용 사례가 조기에 개발되어 설치된다는 확신
  - 사용 사례 사이에 선행관계를 지킬 수 있다.
  - 각 열의 점수의 합은 개발 팀의 작업속도를 초과 하지 않아야 한다.

3.4 조직 계획

- **조직의 구성**
  - 소프트웨어 개발 생산성에 큰 영향
  - 작업의 특성과 팀 구성원 사이의 의사교류 (3 ~ 8명이 적절한 규모)

- **프로젝트의 구조**
  - **프로젝트별 조직**
    - 프로젝트 시작에서 개발 완료까지 전담 팀
  - **기능별 조직**
    - 계획수립 분석팀, 설계 구현 팀, 테스트 및 유지보수 팀
    - Pipeline 식 공정

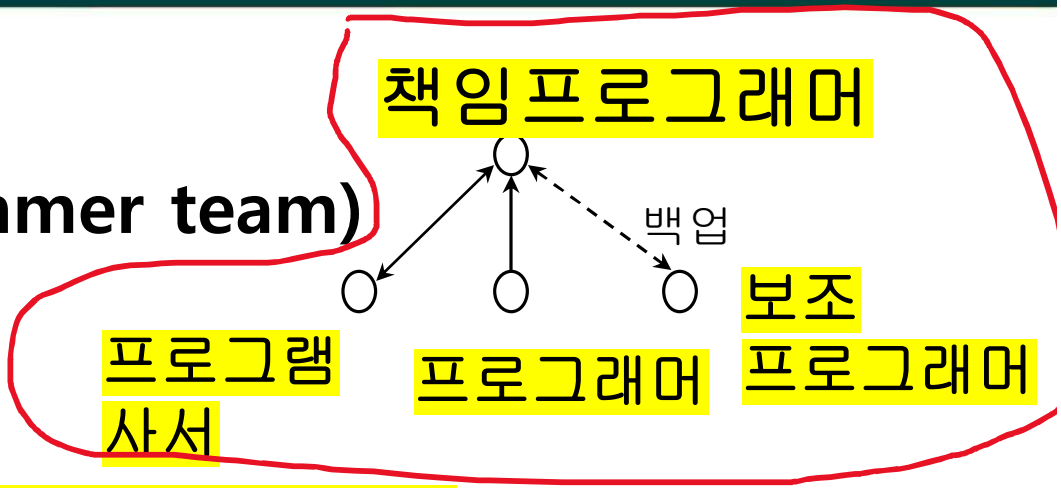
3.4 조직 계획

- **매트릭스 조직**
  - 요원들은 프로젝트 팀과 기능 조직(회사내 부서)에 동시에 관련
  - 필요에 따라 요원을 차출 팀을 구성하고 끝나면 원래의 소속으로 복귀

	real-time programming	graphics	databases	QA	testing
project C	X			X	X
project B	X		X	X	X
project A		X	X	X	X

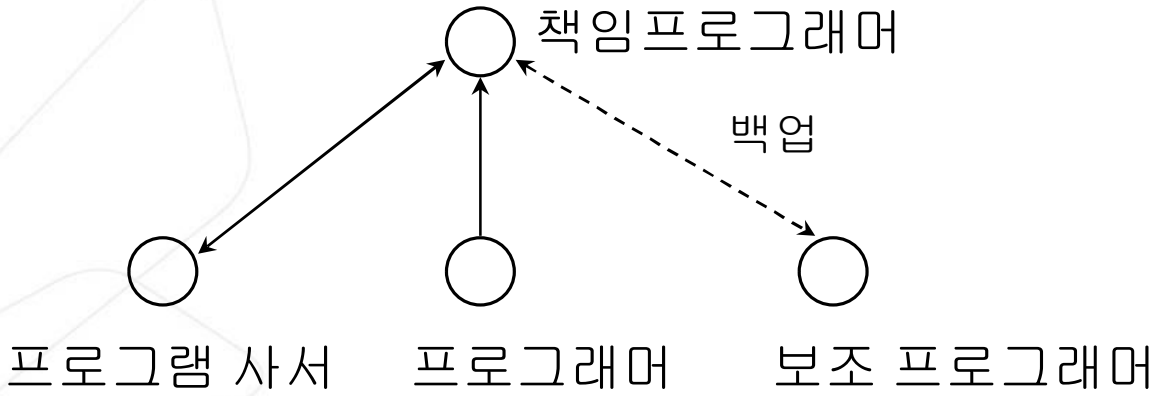
책임 프로그래머 팀 (프로그래밍 팀 구성)

- **의사 결정권이 리더에게 집중**
- **계층적 팀 구조(chief programmer team)**
- **팀원 역할**
  - 외과 수술 팀 구성에서 따옴
  - 책임 프로그래머: 제품설계, 주요부분 코딩, 중요한 기술적 결정, 작업의 지시
  - 프로그램 사서: 프로그램 리스트 관리, 설계 문서 및 테스트 계획 관리
  - 보조 프로그래머: 기술적 문제에 대하여 상의, 고객/출판/품질 보증 그룹과 접촉, 부분적 분석/설계/구현을 담당
  - 프로그래머: 각 모듈의 프로그래밍



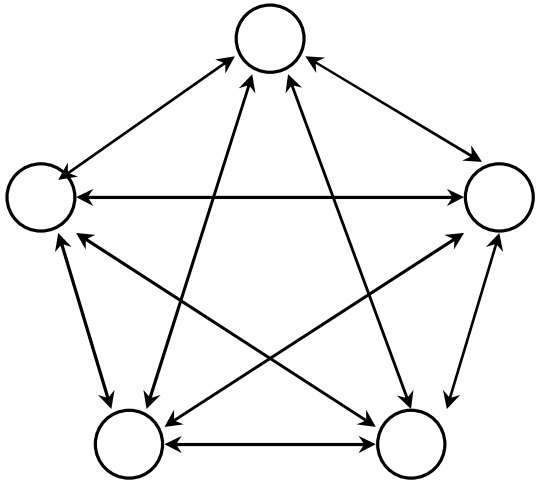
책임 프로그래머 팀

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



에고레스 팀조직 (프로그래밍 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 프로젝트 (복잡, 이해 어려움 문제 해결)에 적합
- **단점**
  - 책임이 명확하지 않은 일이 발생
  - 대규모에 적합하지 않음(의사 결정 지연 가능)

혼합형 팀조직 (프로그래밍 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 되는 프로젝트에 적합**
- **단점**
  - 기술인력이 관리를 담당
  - 의사 전달 경로가 김