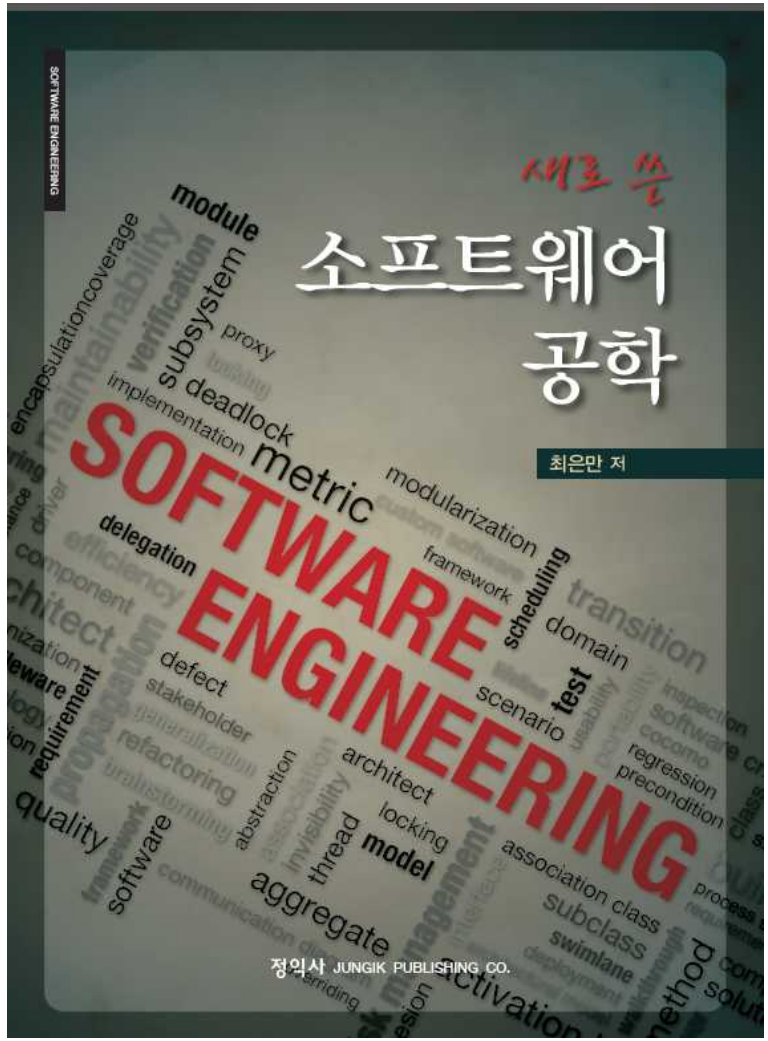
소프트웨어 공학

Lecture #15: 프로젝트 관리와 계획

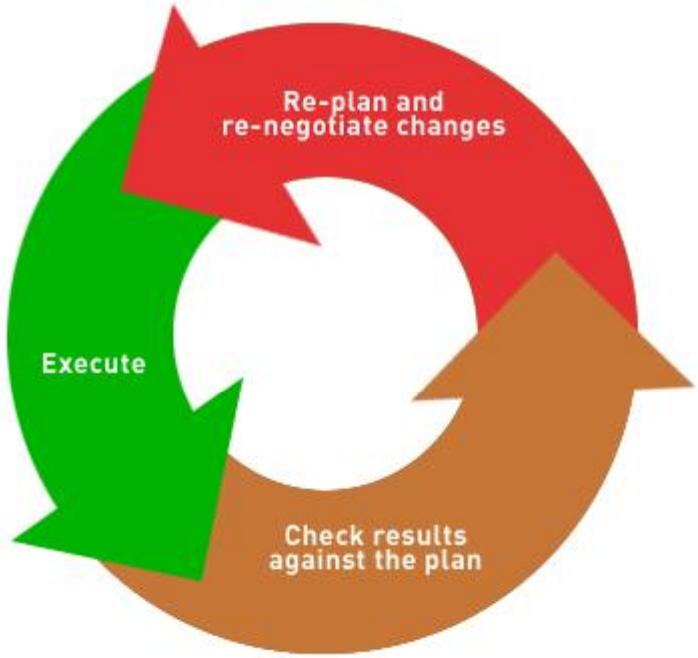
최은만 저

6차 개정판



학습 목표

- 프로젝트 범위
- 노력 추정
- 일정 계획
- 프로젝트 조직
- 위험 관리
- 프로젝트 관리 도구



프로젝트 관리(Management)

- 프로젝트 관리란?

  소프트웨어 프로젝트를

  - 조직하고(organizing)
  - 계획하고(planning)
  - 일정관리(Scheduling) 하는 것이다.

계 획

- 계획의 부재
  - 불확실성
  - 일정의 차질, 경비의 초과, 저품질, 높은 유지보수 비용
  - Risk
  - 프로젝트의 실패
- 소프트웨어 프로젝트 계획 수립

  "소프트웨어 개발 과정과 일정, 비용, 조직, 생산 제품에 대하여 사전에 계획"
  - 문제를 이해하고 정의
  - 필요한 소작업을 정의하고 순서를 결정
  - 일정 예측
  - 비용 예측
  - 위험 분석

  } => 계획서

계 획

- 계획 수립의 결과

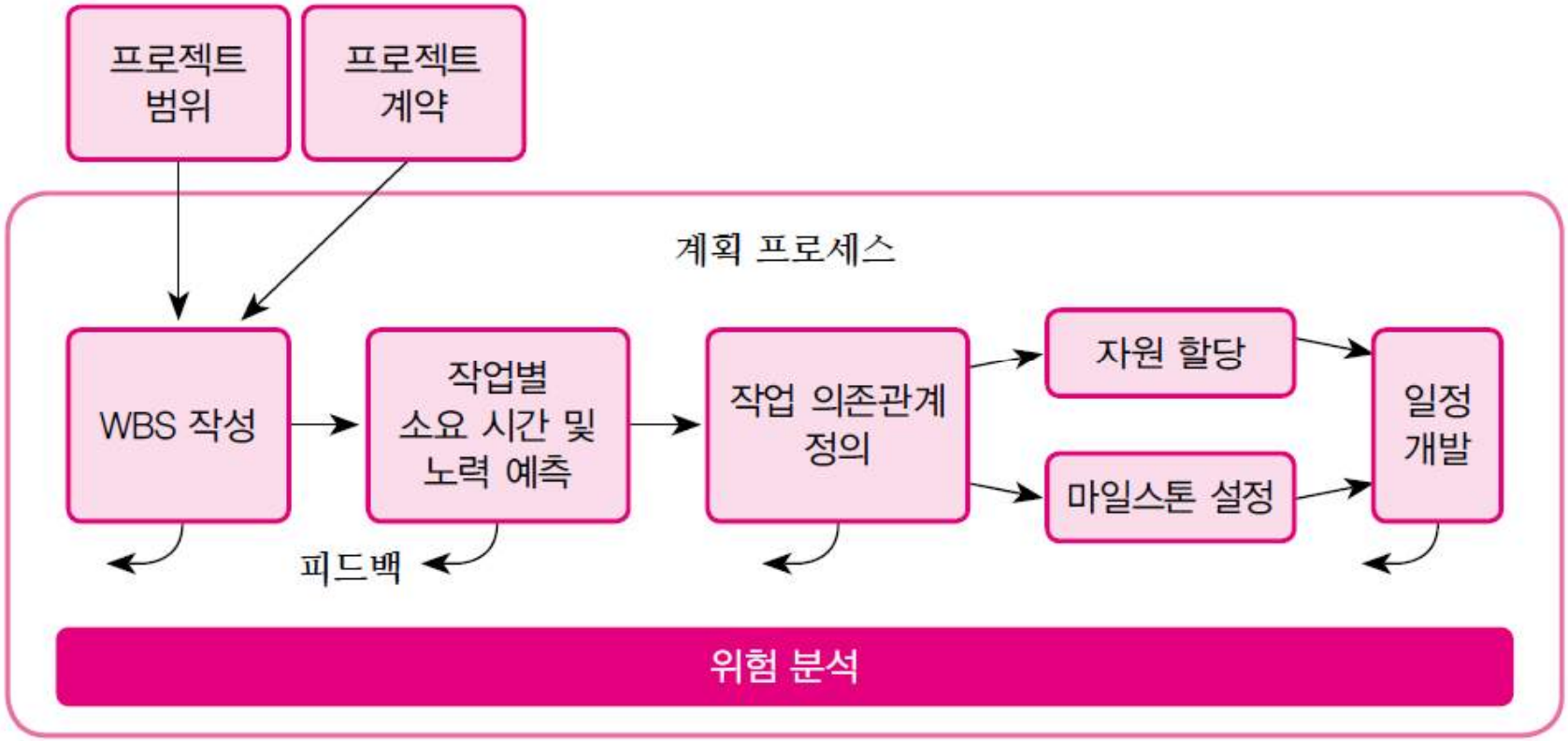
  -> 소프트웨어 개발 계획서

  - 사업관리자, 개발자, 사용자들에게 사업의 범위, 필요 비용, 필요 자원, 개발 일정, 위험 요소 등에 대한 정보를 제공하는 산출문서(deliverable)

- 주의할 점
  - 시스템에 대한 충분한 이해, 그러나 변경의 여지도 있음
  - 현실적, 구체적 계획
  - 득실 관계 저울질
  - 기술적인 측면 고려

프로젝트 일정 계획 작업 과정

- 일정 계획을 위한 과정



3.1 프로젝트 범위

- 소프트웨어 개발 프로젝트를 위한 계획은 대상 업무나 문제의 범위(Scope)를 정하는 것으로 부터 시작

- 문제의 범위를 정의 하기 위하여 먼저 문제의 배경과 응용분야를 잘 이해
  - 사용자와 면담
  - 현장 관찰
  - 실제업무수행
  - 문제 정의

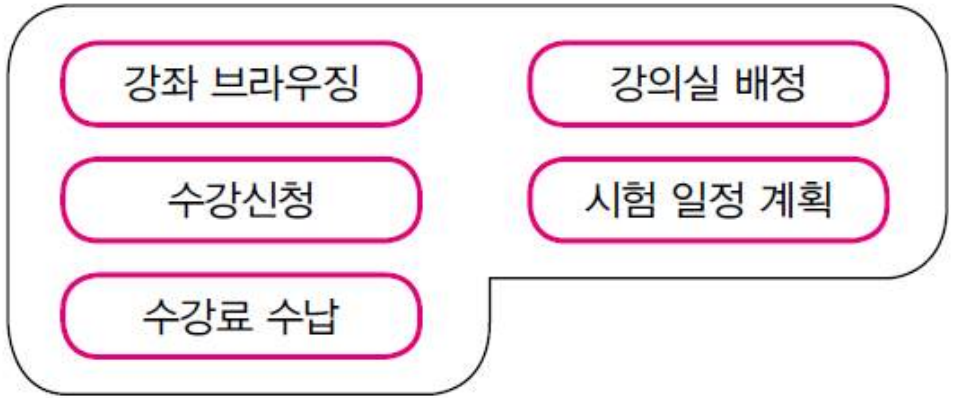
문제 정의

- 대책 수립
  - 신규 시스템의 목표 설정
    - 기능과 우선순위(투자 효과를 분석)
  - 해결 방안 모색(사용자 요구, 개발 여건, 기술적 능력 고려)

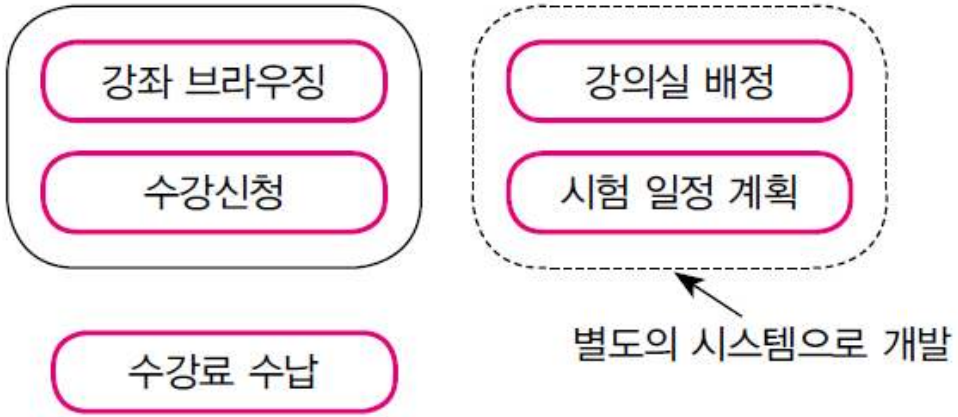
- 시스템 정의
  - 문제의 기술
  - 시스템의 필요성
  - 시스템의 목표
  - 제약 사항
  - 시스템의 제공 기능
  - 사용자의 특징
  - 개발, 운용, 유지보수 환경

문제 범위 정하기

- 수강 신청 시스템
  - 넓은 범위



  - 작은 범위



3.2 노력 추정

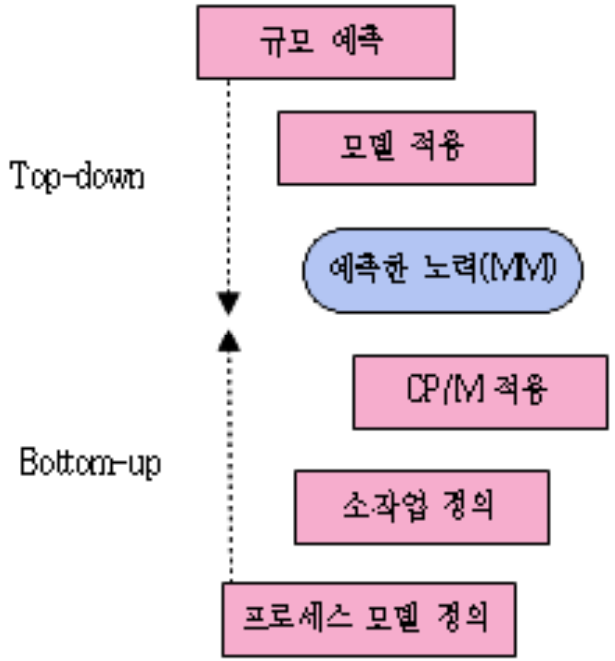
- 소프트웨어 개발 비용 예측
  - 정확한 비용 예측은 매우 어려움
    - 알려지지 않은 요소가 산재
    - 원가의 계산이 어려움
  - 과거의 데이타가 필요
  - 단계적 비용 산정 방법도 사용
- 예산
  - 인건비: MM(인원/월)을 기초
  - 경비: 여비, 인쇄비, 재료비, 회의비, 공공요금
  - 간접 경비: overhead

비용에 영향을 주는 요소

- 제품의 크기
  - 제품의 크기가 커짐에 따라 기하급수로 늘어남
- 제품의 복잡도
  - 응용 : 개발지원 : 시스템 = 1 : 3 : 9
- 프로그래머의 자질
  - 코딩, 디버깅의 능력차
  - 프로그래밍 언어, 응용 친숙도
- 요구되는 신뢰도 수준
- 기술 수준(개발 장비, 도구, 조직능력, 관리, 방법론 숙달)
- 남은 시간
  - Putnam “프로젝트의 노력은 남은 개발 기간의 4제곱에 반비례”

프로젝트 비용을 예측하는 방법

- 상향식
  - 소요 기간을 구하고 여기에 투입되어야 할 인력과 투입 인력의 참여도를 곱하여 최종 인건 비용을 계산
  - 소작업에 대한 노력을 일일이 예측
- 하향식
  - 프로그램의 규모를 예측하고 과거 경험을 바탕으로 예측한 규모에 대한 소요 인력과 기간을 추정
  - 프로그램의 규모
    - LOC
    - 기능 점수



COCOMO 방법

- Boehm이 개발
  - TRW의 2K-32K 정도의 많은 프로젝트의 기록을 통계 분석
- 표준 산정 공식

	노력(MM)	기간(D)
유기형	PM = 2.4*(KDSI)**1.05	TDEV=2.5*(PM)**0.38
반결합형	PM = 3.0*(KDSI)**1.12	TDEV=2.5*(PM)**0.35
내장형	PM = 3.6*(KDSI)**1.20	TDEV=2.5*(PM)**0.32

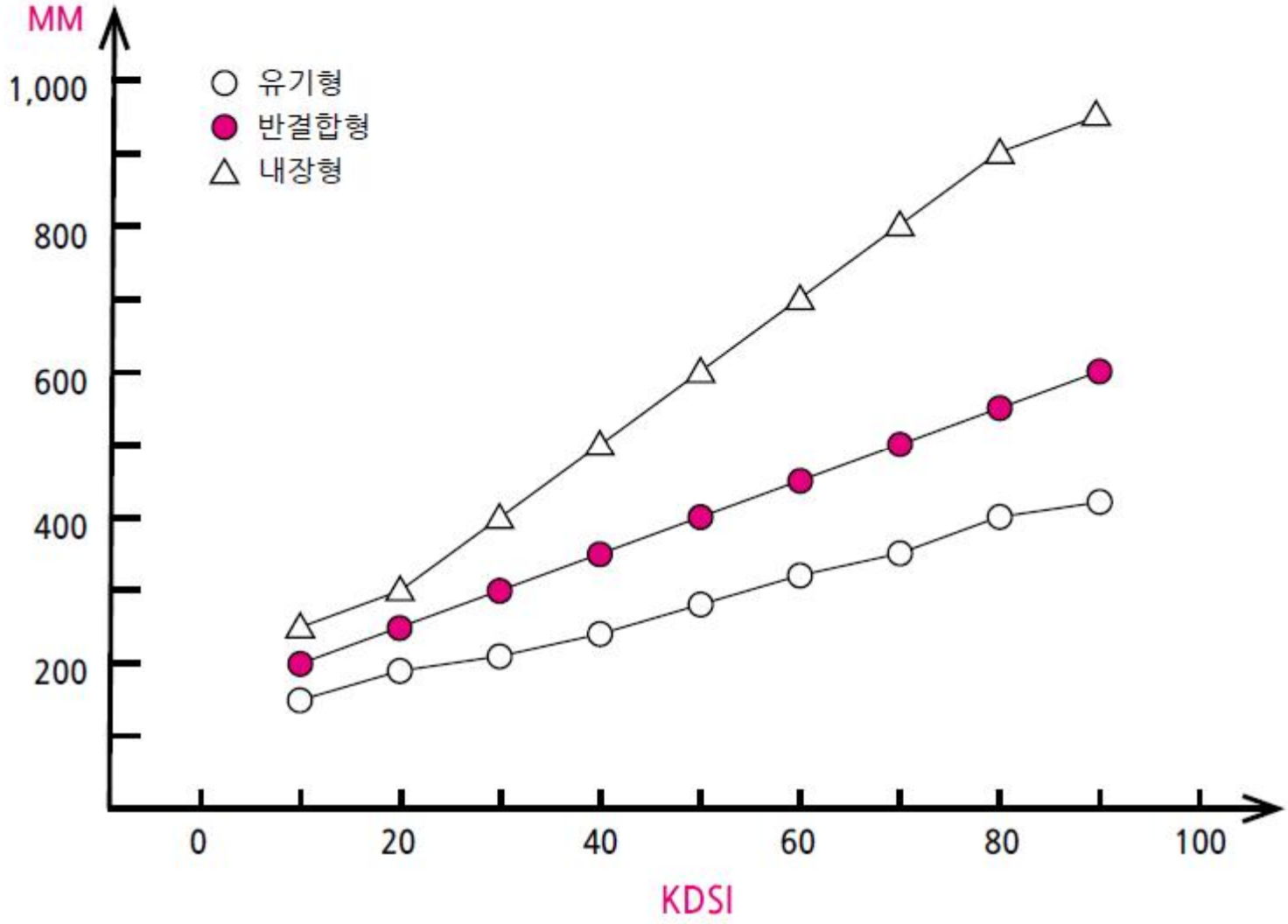
- 예
  - CAD 시스템 예상 규모: 33360 LOC

    PM = 3.0*(KDSI)**1.12 = 3.0*(33.3)**1.12 = 152MM

    TDEV = 2.5*(PM)**0.35 = 2.5*(152)**0.35 = 14.5 M

    N=E/D = 152/14.5 ~ 11 명
  - 보정

COCOMO에 의한 비용 예측



COCOMO-81 방법

모델	내용	기타
기본 COCOMO (Basic COCOMO)	추정된 LOC를 프로그램 크기의 함수로 표현해서 소프트웨어 개발 노력(그리고 비용)을 계산.	S/W 크기와 개발모드
중간급COCOMO (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

중간 COCOMO 방법

	비용 드라이버	비율					
		매우낮음	낮음	보통	높음	매우높음	극히매우높음
제품특성	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
H/W	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	
개인특성	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		
PROJECT 특성	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	

중간 COCOMO 방법

- 모든 노력 승수를 곱한다.
  - 예: $E=EAF * 2.4(32)^{1.05} = EAF * 91$ man-months

- 단점
  - 소프트웨어 제품을 하나의 개체로 보고 승수들을 전체적으로 적용시킴
  - 실제 대부분의 대형 시스템은 서로 상이한 서브 시스템으로 구성되며 이중 일부분은 Organic Mode이고 다른 부분은 Embedded Mode인 경우도 있다.

COCOMO II

- 1995년에 발표
- 소프트웨어 개발 프로젝트가 진행된 정도에 따라 세가지 다른 모델을 제시
  - 1 단계: 프로토타입 만드는 단계
    - 화면이나 출력 등 사용자 인터페이스, 3 세대 언어 컴포넌트 개수를 세어 응용 점수(application points)를 계산
    - 이를 바탕으로 노력을 추정
  - 2 단계: 초기 설계 단계
    - 자세한 구조와 기능을 탐구
  - 3 단계: 구조 설계 이후 단계
    - 시스템에 대한 자세한 이해

COCOMO II 세 가지 단계

비교대상	단계 1: 응용합성 (프로토타이핑)	단계 2: 초기 설계	단계 3: 설계 이후
크기	응용 포인트	기능 포인트(FP)와 언어 종류	FP와 언어 LOC
재사용	모델에 포함됨	LOC를 다른 변수의 함수로 사용	LOC를 다른 변수의 함수로 사용
요구변경	모델에 포함됨	변경 비율이 비용승수로 반영됨	변경 비율이 비용승수로 반영됨
유지보수	응용 포인트 연평균 변경 비율 (ACT)	ACT, 이해력, 친밀성의 함수	ACT, 이해력, 친밀성의 함수
노력 예측 공식 ($E=bS^C$) 에서 C의 값	1.0	선행작업, 적응도, 초기 설계, 위험제거, 팀 결집력, SEI 프로세스 성숙도에 따라 0.91 ~ 1.23	선행작업, 적응도, 초기 설계, 위험제거, 팀 결집력, SEI 프로세스 성숙도에 따라 0.91 ~ 1.23

COCOMO II 세 가지 단계

비교대상	단계 1: 응용합성 (프로토타이핑)	단계 2: 초기 설계	단계 3: 설계 이후
프로덕트 비용승수	없음	복잡도, 재사용 요구도	신뢰도, 데이터베이스 규모, 문서화 요구정도, 재사용 요구도, 제품 복잡도
플랫폼 비용승수	없음	플랫폼 난이도	실행시간 제약, 기억공간 제약, 가상기계
인력 비용 승수	없음	개인 능력과 경험	분석 능력, 응용 경험, 프로그래머 능력, 프로그래머 경험, 언어 및 도구사용 경험, 연속성
프로젝트 비용승수	없음	개발 기간, 개발 환경에 대한 요구	소프트웨어 도구 사용, 개발 기간, 여러 사이트 개발 요구

기능 점수 방법

- 기능 점수(function points)
  - 정확한 라인수는 예측 불가능
  - 입력, 출력, 질의, 화일, 인터페이스의 개수로 소프트웨어의 규모를 나타냄
  - 각 기능에 가중값(표 2.6)
  - 기능 점수 1을 구현하기 위한 LOC
    - 어셈블리 언어(324), C언어(150), Pascal(91), Ada(71), APL(32)
- 복합 가중값을 이용한 기능점수 산출(표 2.7)
- 총 라인수 = FP * 원하는 언어의 1점 당 LOC
- 개발 노력 = 총라인수 / 생산성(LOC/MM)

기능 점수 기본 개념

- 기능 점수는 총 기능 점수(Gross Function Point)와 처리 복잡도 보정 계수(Processing Complexity Adjustment)를 곱한 것이다.

$$FP = GFP \times PCA$$

- 기능 점수는 구현되는 언어에 관계없는 메트릭이다.
- 기능 점수 방법은 모든 항목에 일률적인 가중치가 적용되므로 문제가 있을 수 있다.

기능 점수 구하는 방법

1. 다섯 가지 기능 분야에 해당되는 개수를 파악
2. 다섯 각 기능에 대한 복잡도(단순, 중간, 복잡)를 결정
3. 각 기능 분야의 개수와 복잡도 가중치를 곱하여 총 기능 점수(GFP)를 구한다.

$$GFP = \sum_{i=1}^{5}(Count_i \times Complexity_i)$$

4. 14개의 질문을 이용하여 각 처리 복잡도의 정도에 따라 0에서 5까지 할당한다.
5. 처리 복잡도 보정계수(PCA)를 다음 식을 이용하여 구한다.

$$PCA = 0.65 + 0.01 \sum_{i=1}^{14} PC$$

6. 다음 식에 넣어 기능 점수를 구한다.

$$FP = GFP \times PCA$$

기능 점수를 이용한 노력 추정

- 파악된 기능
  - 사용자 입력 = 10개, 사용자 출력 = 5개, 사용자 질의 = 8개, 자료 파일 = 30개, 외부 인터페이스 = 4개, 복잡도는 모두 단순
- 처리 복잡도
  - 신뢰도 높은 백업, 사용 친근성은 매우 높이 요구되며 나머지는 보통
- 생산성
  - 60 FP/week

- 해답
  - [표 3.5]에 대입하여 GFP를 구한다.

    $GFP = 10 \times 3 + 5 \times 4 + 8 \times 3 + 30 \times 7 + 4 \times 5 = 304$ FP
  - 처리 복잡도 보정 계수 구하면

    $PCA = 0.65 + 0.01(12 \times 3 + 2 \times 3) = 1.11$
  - FP를 보정

    $FP = GFP \times PCA = 304 \times 1.11 = 337.44$ FP
  - 추정 노력(E) = FP / 생산성 = 337.44 / 60 = 5.624 persons-week

국내 기능 점수 산정 가이드

- 정보통신연구진흥원의 소프트웨어 공학에서 산정 기준을 제시[소프트웨어공학 센터, 2010]
- 산정 기준의 큰 틀은 COCOMOII의 초기 설계 모델을 따른다.
  - 외부 입력(External Input)
  - 외부 출력(External Output)
  - 내부 논리 파일(Internal Logical File)
  - 외부 인터페이스 파일(External Interface File)
  - 외부 조회(External Query)

3.3 일정 계획(Scheduling)

- 일정 계획

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

  - 개발 모형 결정
  - 소작업, 산출물, 이정표 설정

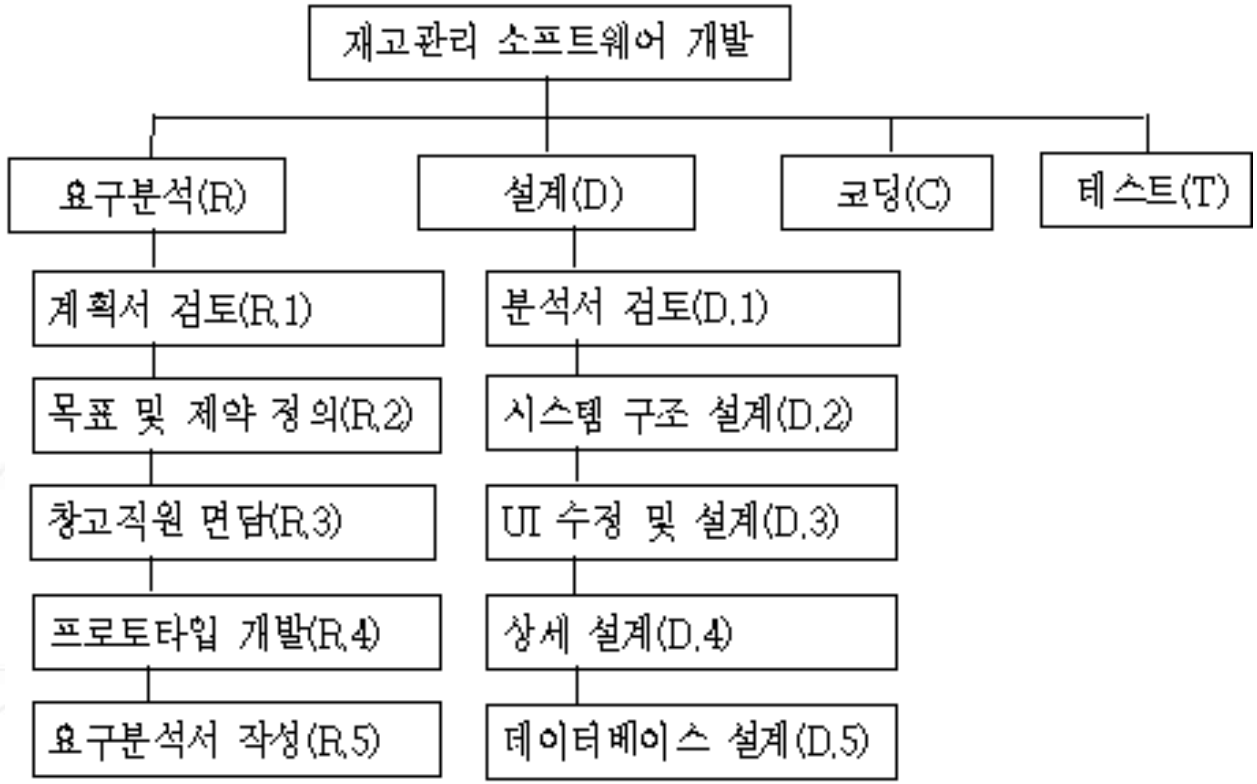
- 작업 순서
  - 작업분해(Work Breakdown Structure)
  - CPM 네트워크 작성
  - 최소 소요 기간을 구함
  - 소요 MM, 기간 산정하여 CPM 수정
  - 간트 차트로 그림

작업 분해(Decomposition)

- 작업 분해

  프로젝트 완성에 필요한 activity를 찾아냄

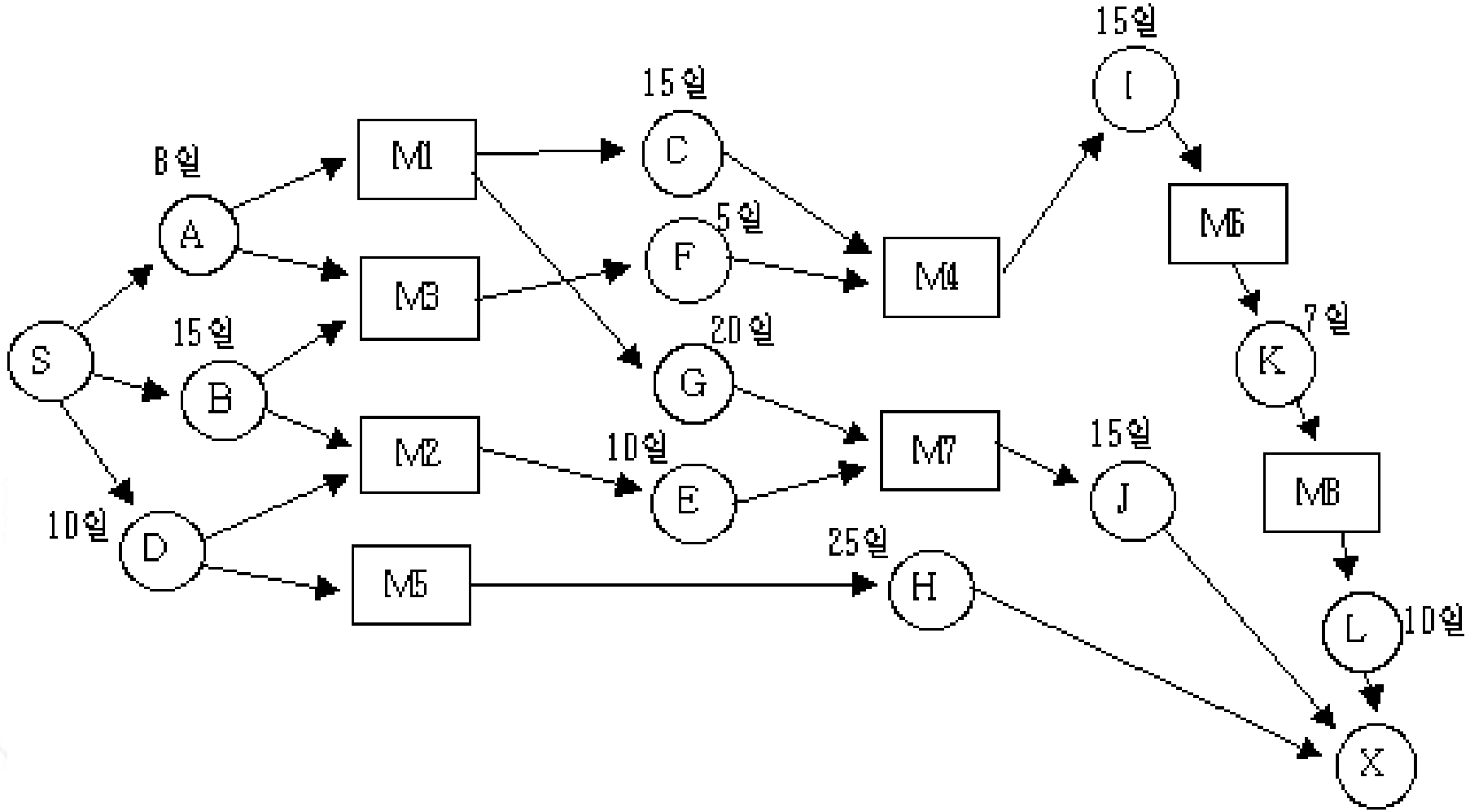
- Work Breakdown Structure
  - 계층적 구조



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



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

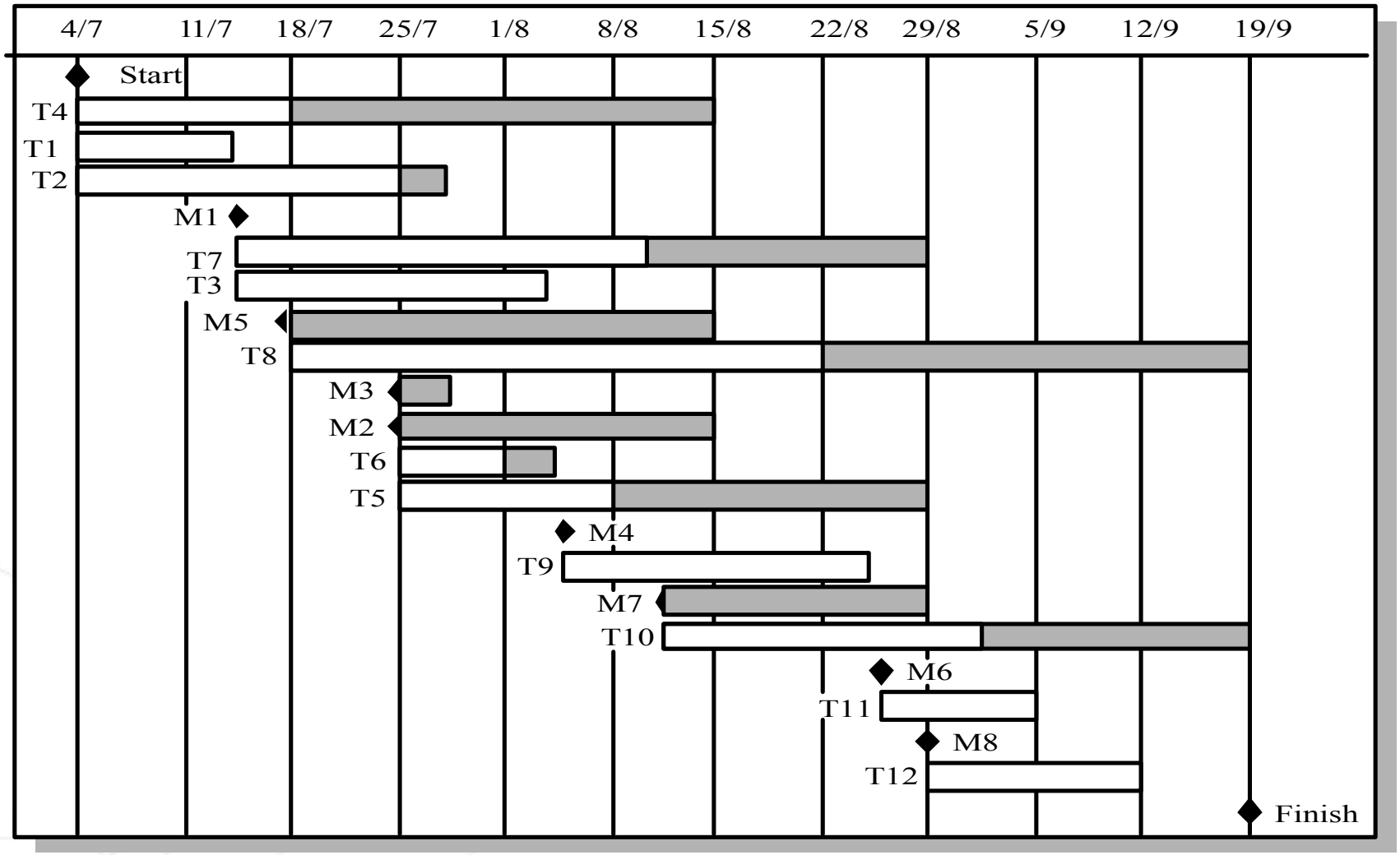
CPM 네트워크

- 장점
  - 관리자의 일정 계획 수립에 도움
  - 프로젝트 안에 포함된 작업 사이의 관계
  - 병행 작업 계획
  - 일정 시뮬레이션
  - 일정 점검, 관리
- 관리에 대한 작업도 포함 가능
- 작업 시간을 정확히 예측할 필요
- 소프트웨어 도구
  - MS-Project, MS-Works 등

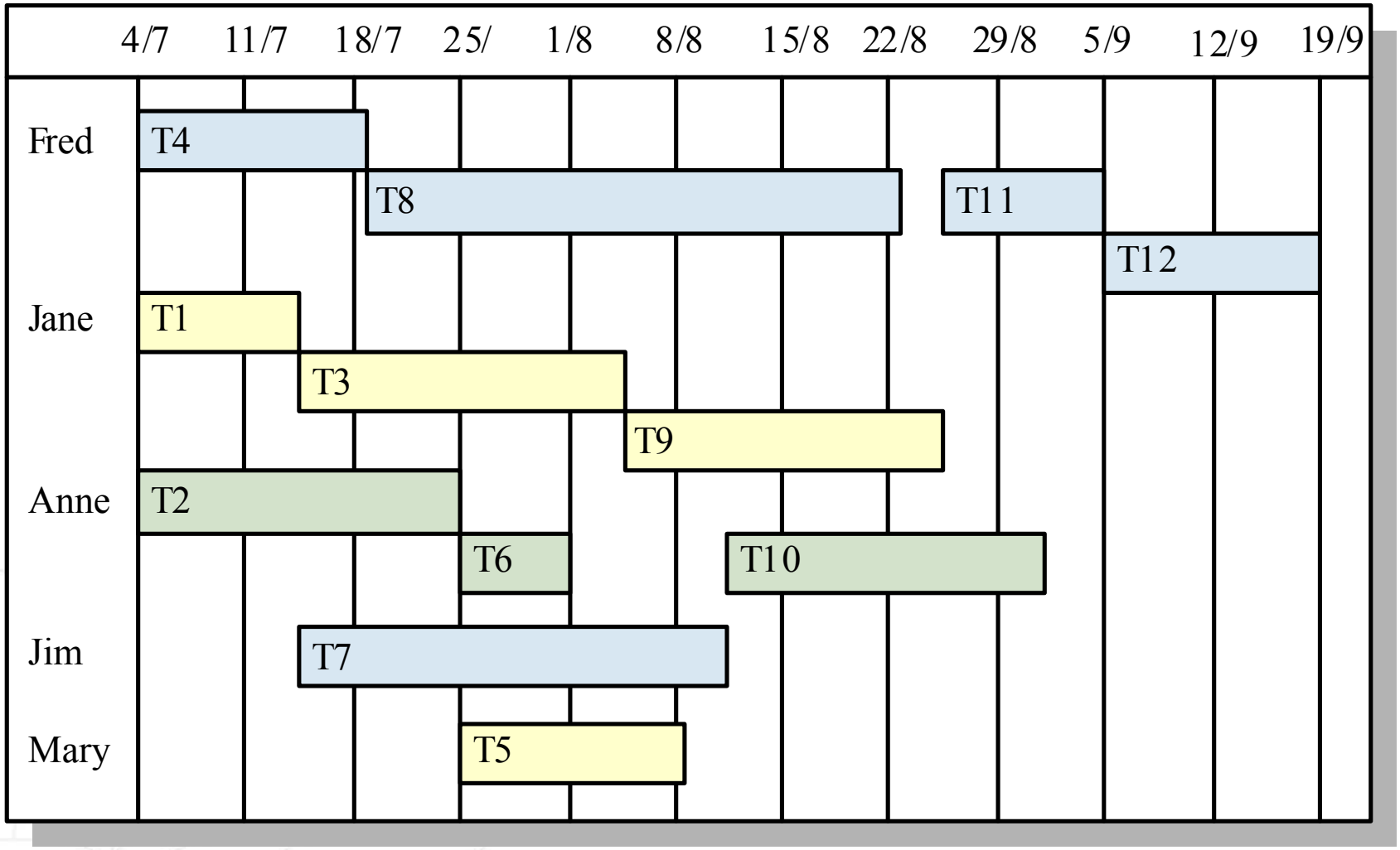
간트 차트

- 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
T11
M8
T12
Finish

Staff Allocation



애자일 계획

- 애자일 프로세스의 일정 계획

- 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

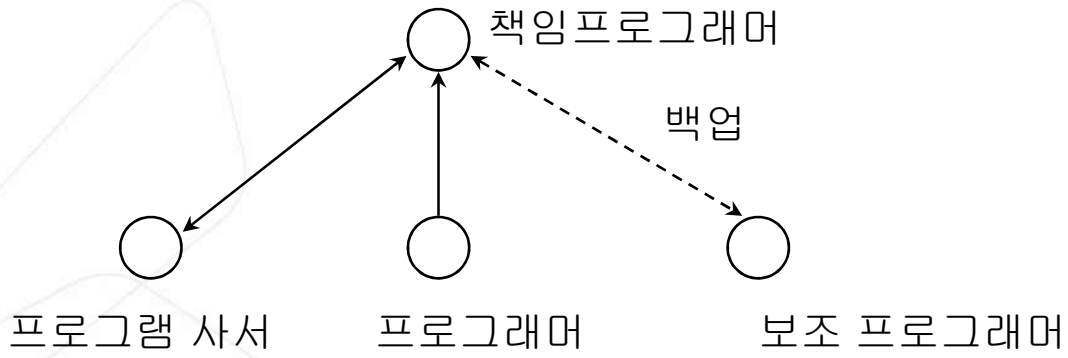
- 조직의 구성
  - 소프트웨어 개발 생산성에 큰 영향
  - 작업의 특성과 팀 구성원 사이의 의사교류
- 프로젝트의 구조
  - 프로젝트별 조직
    - 프로젝트 시작에서 개발 완료까지 전담 팀
  - 기능별 조직
    - 계획수립 분석팀, 설계 구현 팀, 테스트 및 유지보수 팀
    - Pipeline 식 공정
  - 매트릭스 조직
    - 요원들은 고유 관리 팀과 기능 조직에 동시에 관련
    - 필요에 따라 요원을 차출 팀을 구성하고 끝나면 원래의 소속으로 복귀

책임 프로그래머 팀

- 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

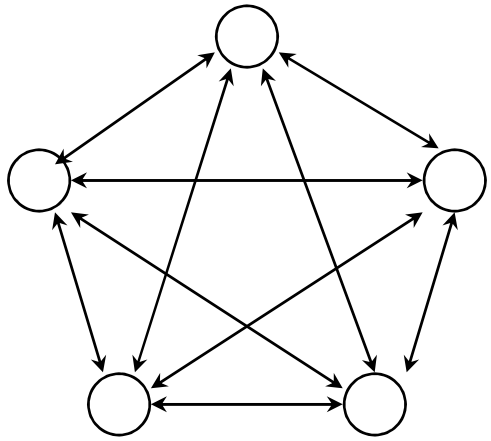
책임 프로그래머 팀

- 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

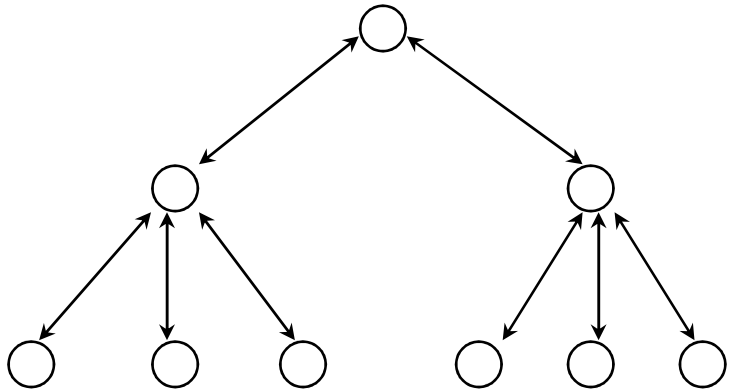
- 민주주의 식 의사결정
  - 서로 협동하여 수행하는 비이기적인 팀(Ego-less)
  - 자신이 있는 일을 알아서 수행
  - 구성원이 동등한 책임과 권한
- 의사 교환 경로



- 특징
  - 작업 만족도 높음
  - 의사 교류 활성화
  - 장기 프로젝트에 적합
- 단점
  - 책임이 명확하지 않은 일이 발생
  - 대규모에 적합하지 않음(의사 결정 지연 가능)

혼합형 팀조직

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



3.5 위험 관리

- 위험 관리의 목적은 위험이 발생되었을 때의 영향을 줄이는 것이다.
- Boehm이 제안한 위험 관리[Boehm, 1991]
  - 위험파악
  - 위험분석
  - 위험 우선순위 정하기
  - 위험 해결
  - 위험 모니터링

위험 파악(1)

위험 요소	위험 관리 기법
1.인력 부족	- 유능한 인력모집, 팀 구성, 요원 배치, 교차-교육, 유능 인력 사전 확보
2.비현실적 일정 및 예산	- 더 자세한 비용, 일정 예측, 원가 분석, 점증적 개발, 소프트웨어 재사용 요구를 줄임
3.잘못된 기능의 소프트웨어 개발	- 사용자 회람, 프로토타이핑, 사용자 지침서를 조기에 작성, 조직 분석, 직능 분석
4. 잘못된 인터페이스의 개발	- 프로토타이핑, 시나리오, 태스크 분석, 사용자 분류(기능, 스타일, 업무)
5. 과포장(필요 지않을 좋은 기능을 추가)	- 요구 삭감, 프로토타이핑, 비용-수익 분석, 원가 분석

위험 파악(2)

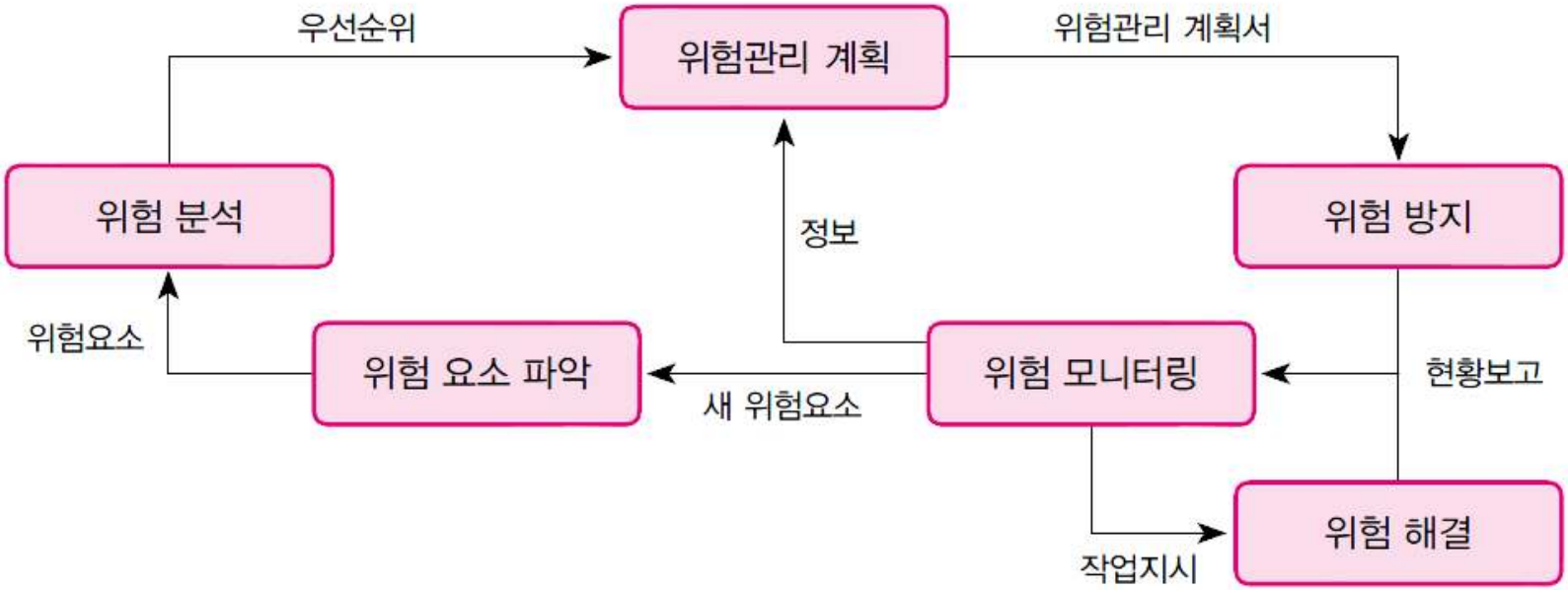
위험 요소	위험 관리 기법
6. 계속적인 요구 변경	- 최대 변경 상한선, 정보 은닉, 점증적 개발(다음 버젼까지 변경을 연기)
7. 외부 모양의 빈약	- 벤치마킹; 검사; 대조 확인; 성숙도 분석
8. 외부 기능의 빈약	- 대조 확인; 사전 검증; 설계 경연; 팀 작업
9. 실시간 성능의 빈약	- 시뮬레이션, 벤치마킹, 모델링, 프로토타이핑, 튜닝
10. 기술적 취약	- 기술 분석, 비용-수익 분석, 프로토타이핑; 점검

위험 분석과 우선순위 정하기

- 위험 분석은 각 위험에 대한 피해 정도, 위험 해결 방법, 이에 대한 비용들을 결정하는 것이다.
  - 손실 발생 확률
  - 손실 발생 규모
  - 위험 노출(exposure)

위험 해결과 모니터링

- 위험 해결은 위험 관리 계획에 명시된 위험을 줄이는 기법을 구현하고 실행하는 것이다.



위험 관리 프로세스

3.6 계획서 작성

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

계획서 작성

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

계획서 작성

8 개발 환경

9 성능 시험 방법

10 문서화

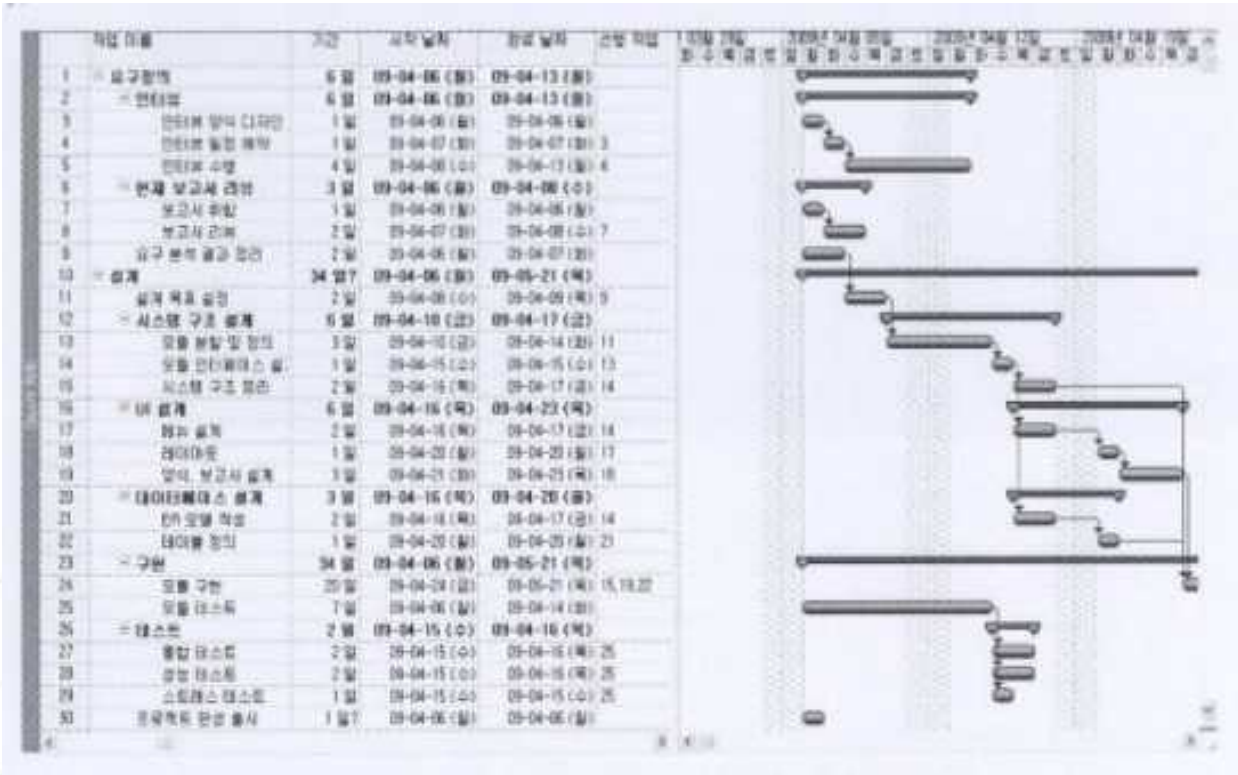
11 유지보수

12 설치, 인수

13 참고문헌 및 부록

도구

- 프로젝트 계획과 관리 도구
  - 노력 추정을 위한 도구
  - 일정 계획 및 진도 관리 도구
  - 문서 및 버전 관리 도구



Questions?