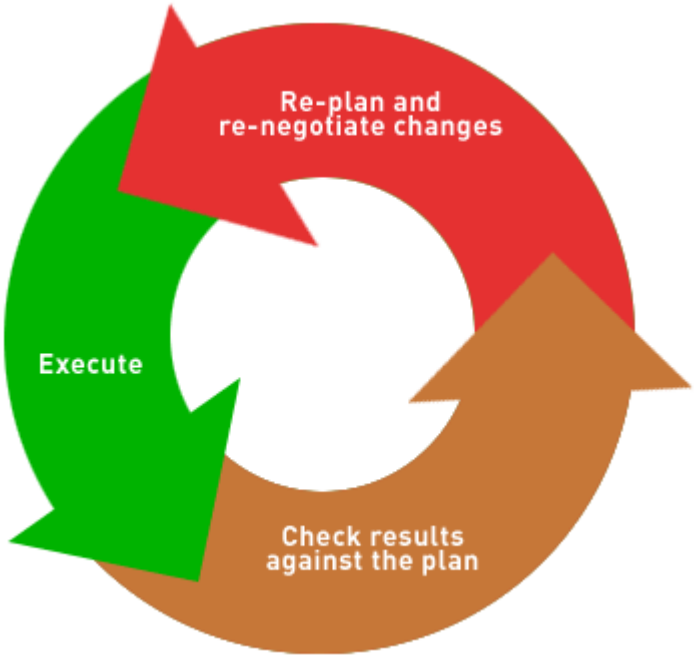
Chap 03 프로젝트 관리와 계획

목 차



프로젝트 관리(Management)

- 프로젝트 관리
- 소프트웨어 프로젝트를
  - 조직하고(organizing)
  - 계획하고(planning)
  - 일정관리(Scheduling) 하는 것이다.

프로젝트 관리(Management)

- 프로젝트 관리의 목적
  - 작업 수행에 필요한 여러 가지 자원, 인력, 비용, 재료, 기술 등을 가장 효과적으로 사용하여 프로젝트의 목표를 달성하는 것

- 관리의 어려움
  - 개발 대상이 눈에 보이지 않는다
  - 소프트웨어 분야의 기술 발전은 매우 빠르다
  - 소프트웨어 분야는 조직마다 프로세스가 다르다

프로젝트 관리 활동

- 관리 활동의 요소
  - 계획
  - 조직
  - 모니터링
  - 조정

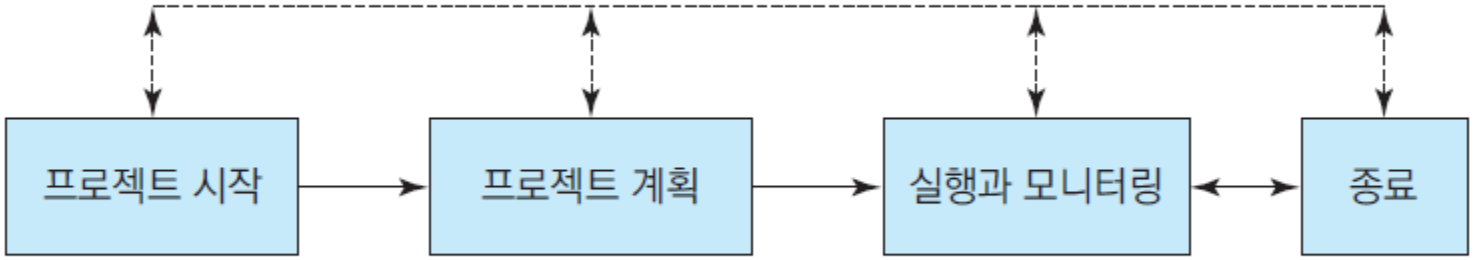


그림 3.2 프로젝트 관리 프로세스

3.1 프로젝트 시작

- 목표를 세우고 가치와 리스크를 이해
- 결정 요인
  - 프로젝트가 제공할 가치
  - 프로젝트와 관련된 리스크
- 가치를 평가하는 방법
  1) 투자 회수 기간
  2) ROI(Return of Investment)
  3) 순수 현재 가치
  4) 평가표
  5) SWOT

리스크와 타당성

- 위험 요인
  - 자원, 현재 사용량과 가용성, 예상 사용량과 가용성, 프로젝트의 우선 순위 및 중요도
  - 시간
  - 기술적 어려움
- 타당성 분석

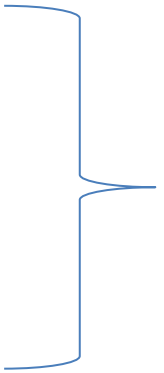
1. SOW(Statement Of Work) – 프로젝트가 성취하여야 할 일. 작업(과업)지시서
2. 비즈니스 목표(가치) - 프로젝트의 결과물
3. 예산 – 비용과 수익의 요약
4. 프로젝트 일정 – 대략적인 일정
5. 프로젝트 리스크 – 위험 요소
6. 대안 – 구축, 구매 등의 방법
7. 평가 – 프로젝트 가치에 대한 평가 결과

3.2 프로젝트 계획과 스케줄링

- 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



3.2 프로젝트 계획과 스케줄링

- 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

3.2 프로젝트 계획과 스케줄링

- 초기 계획
  - 목표 설정
  - 일정 정의
  - 비용 추정

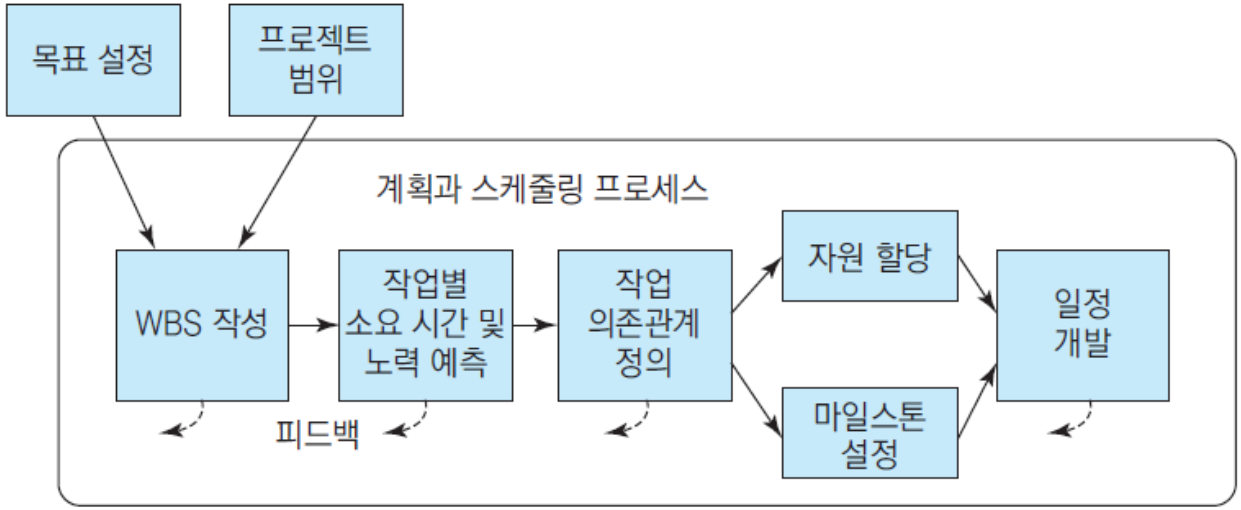
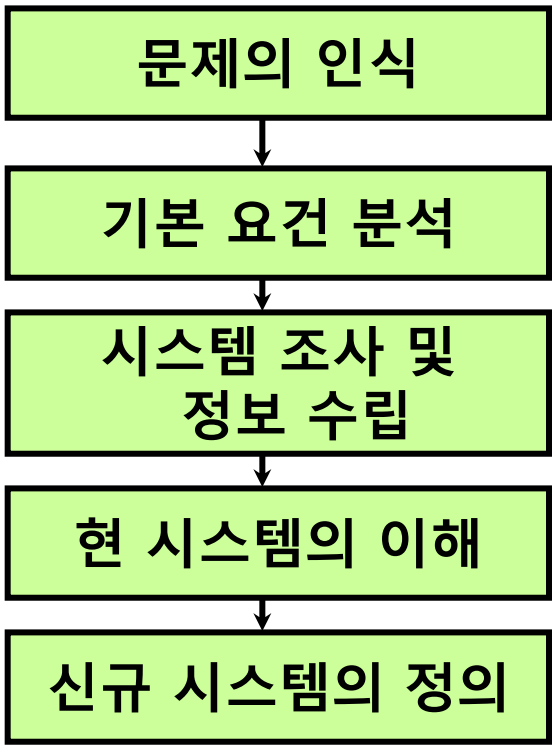


그림 3.4 프로젝트 일정 계획

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

프로젝트 범위



- 문제 범위와 원인 파악
- 문제를 둘러싼 조직, 제도, 시설, 인원, 기술에 관한 현황 파악
- 현재의 시스템 조사, 업무 흐름 정책 등을 파악
- 면담과 서류로 심층 분석
  (고객 상담, 현업의 분석, 작업의 체험)
- 목표 시스템의 정의

AS-IS -> TO-BE

문제정의

- 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

타당성 분석(Feasibility Analysis)

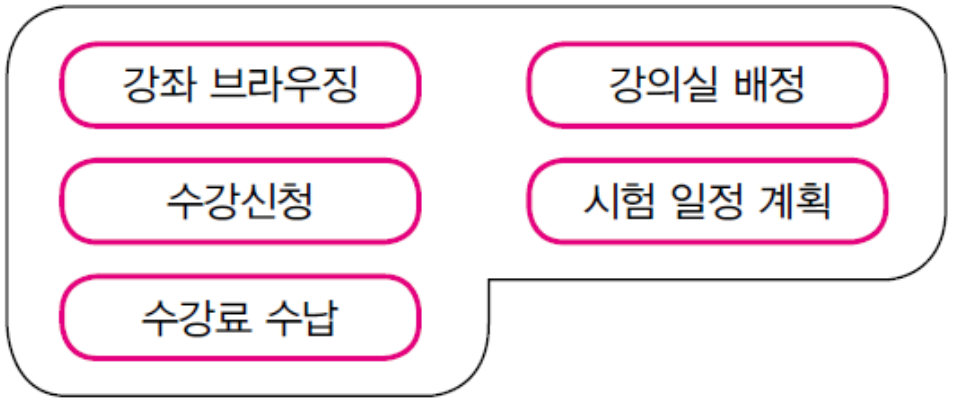
- 경제적 타당성
  - 투자효율성
  - 시장성
  - 비용과 수익의 비교
- 법적 타당성
  - 사용도구들의 법적 권한
  - 시장, 관행들에 대한 조사

타당성 분석(Feasibility Analysis)

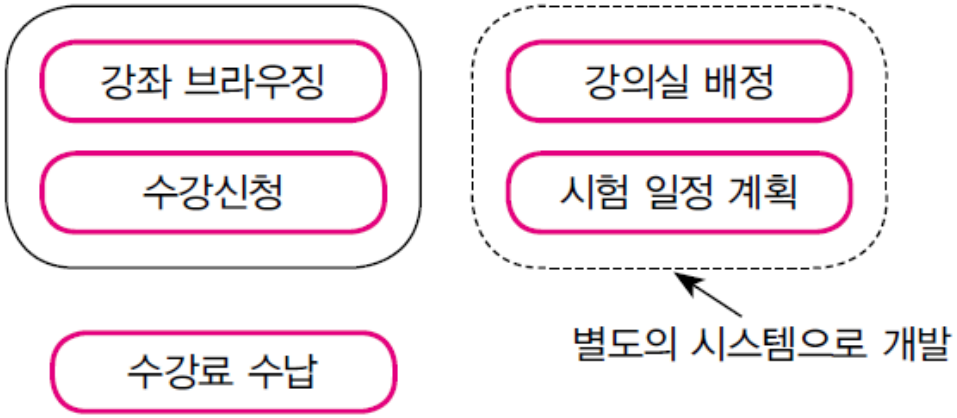
- 기술적 타당성(사용자 요구기능 및 성능 vs. 제공 가능성)
  - 사례연구
  - 실패사례연구
  - 모의실험
  - 프로토타이핑

프로젝트 범위 정하기

- 수강 신청 시스템
  - 넓은 범위



  - 작은 범위



WBS

- 개발 팀이 프로젝트 목표를 달성하고 결과물을 산출하기 위하여 수행하여야 할 작업을 계층적으로 분할한 것

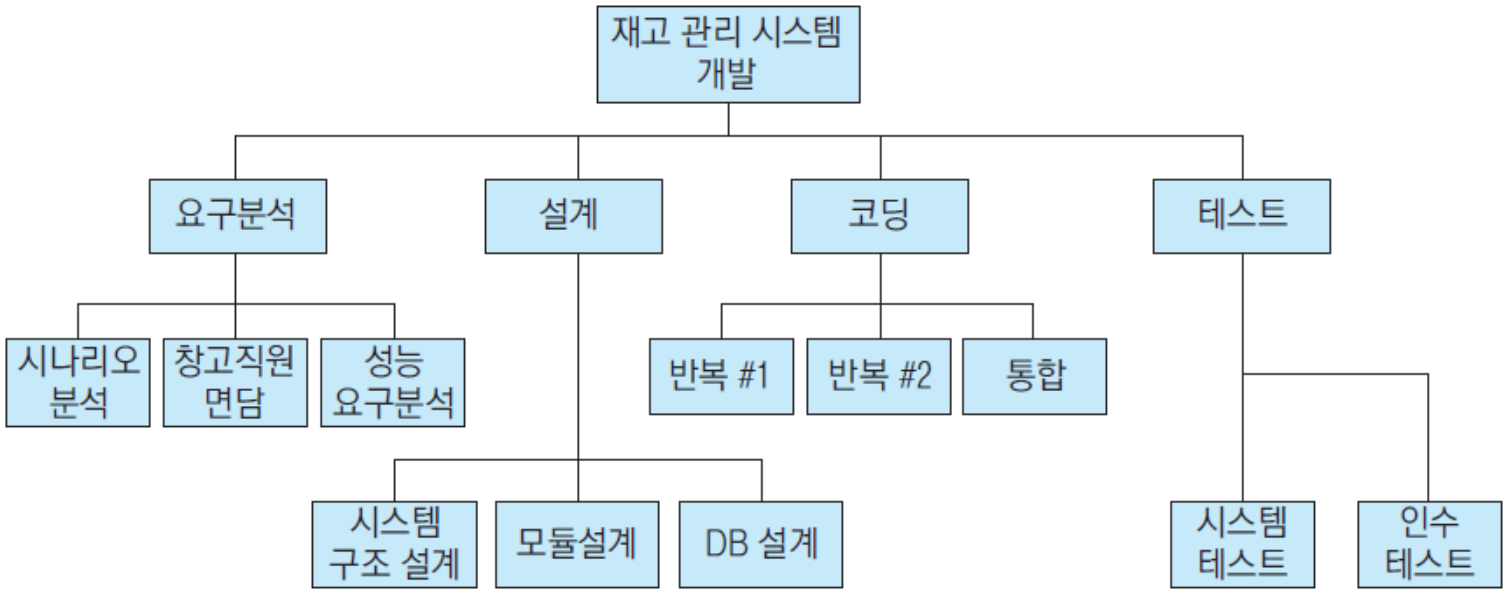


그림 3.6 재고 관리 시스템 개발을 위한 WBS

스케줄링

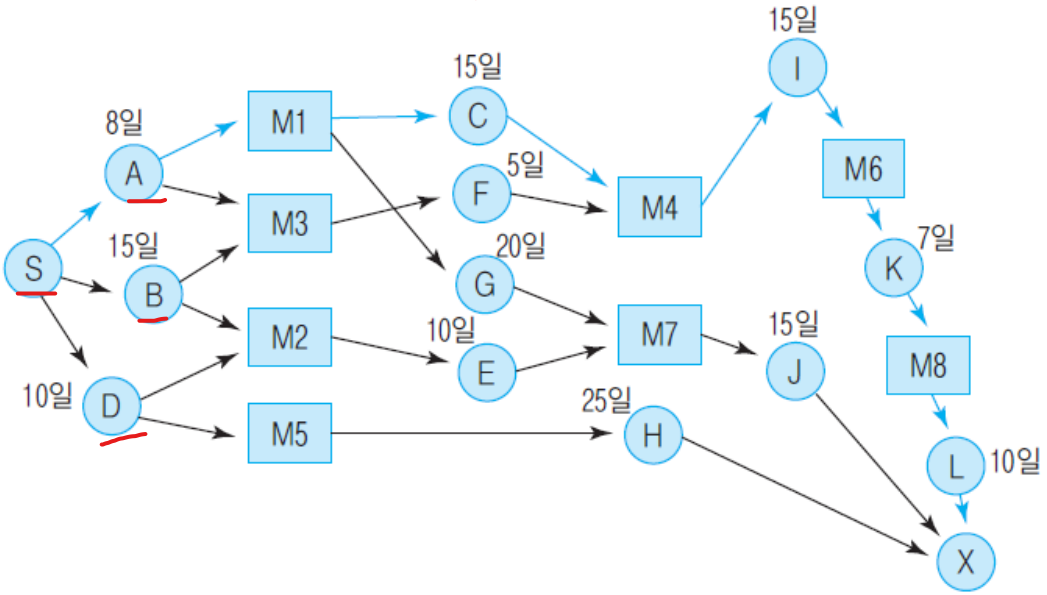
- WBS를 기초로 하여 일정을 정의하는 것
  1) 작업 사이의 의존 관계 파악
  2) CPM(Critical Path Method) 방법을 이용한 여유 시간 계산
  3) 소요 자원의 할당

표 3.1 작업 의존 관계

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

CPM 네트워크

<마이스턴> :



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

CP/M 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

프로젝트 일정표

- 간트 차트
  - 사용목적에 따라 크게 두 가지로 구분
    - 프로젝트 현황파악 : 프로젝트에 대한 Task별 일정, 담당자, 진척도 등을 파악하기 위하여 사용
    - 리소스 현황파악 : 각종 자원이나 사람 등의 리소스에 대하여 시간축을 기준으로 이의 활용된 현황을 파악하기 위하여 사용
  - 소작업별로 작업의 시작과 끝을 나타낸 그래프
  - 예비시간을 보여줌
  - 계획 대비 진척도를 표시

자원 할당과 간트차트

- 일반 태스크(Task) 간트차트
  - 가장 일반적이고 간단한 형태의 간트차트로 각 개별 Task명, 담당자, 시작일, 종료일을 중심으로 태스크의 진행상황을 바(Bar)형태로 한눈에 파악할 수 있도록 표현

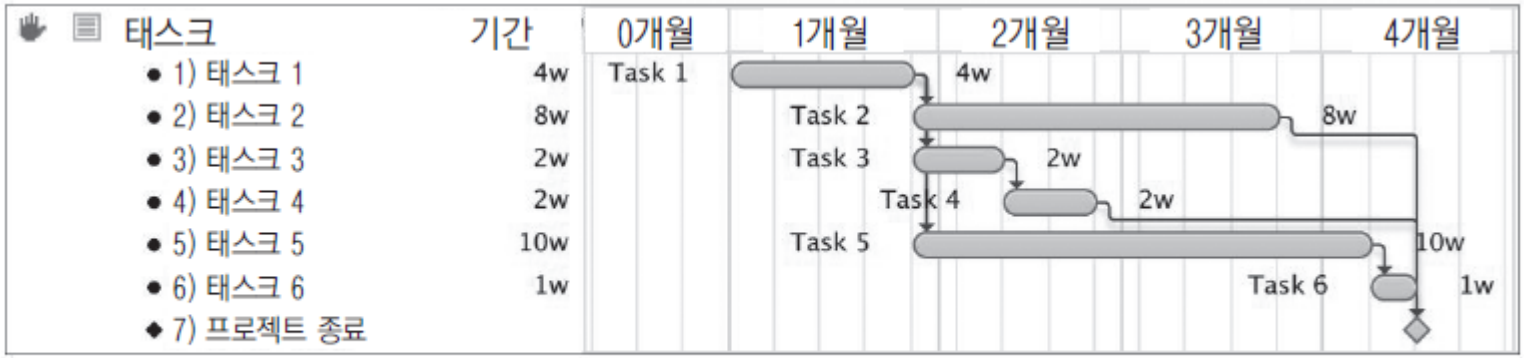


그림 3.8 간트 차트

자원 할당과 간트차트

- 일반 리소스(Resource) 간트차트
  - 회사의 다양한 리소스(사람, 자원등)에 대하여 휴가, 업무 진행도, 자원 활용현황 등의 개별항목을 비교할 수 있음

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**		

그림 3.9 소요 자원의 할당

예산 계획

- 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

비용예측 기법

- 노력(effort)과 자원(resource)과 기간(duration)의 관계

$$D = E/M$$

- 비용 예측의 중요한 변수는 투입되는 엔지니어의 인원수와 작업 기간
- 비용 예측 기법
  - 전문가 판단
  - PERT(Program Evaluation and Review Technique)
  - 알고리즘식 방법

$$T_e = \frac{a + 4m + b}{6} \qquad \sigma = \frac{b - a}{6}$$

Te : 평균기대치 , σ : 표준편차, m: 최빈치 (Most likely Time)
a: 낙관치 (Optimistic Time) b: 비관치 (Pessimistic Time)

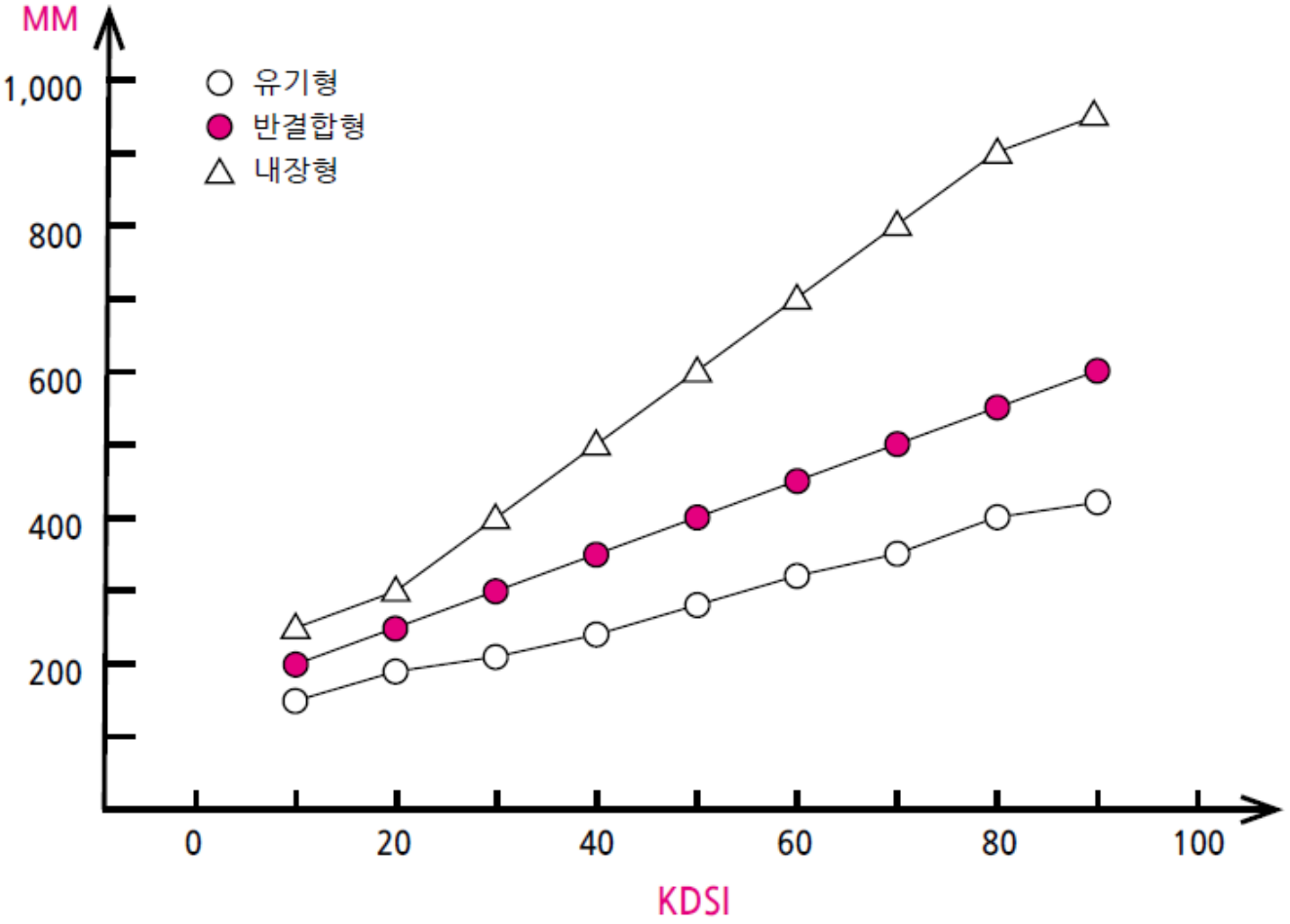
COCOMO-81

- Boehm이 개발
  - TRW의 2K-32K 정도의 많은 프로젝트의 기록을 통계 분석
- 규모를 기반으로 하는 수학적 공식 사용
- 개발 소프트웨어의 유형이나 팀, 프로젝트 프로세스, 등 프로덕트에 영향을 주는 요인을 고려
- 노력 = A × (Size)^{B} × M

표 3.2 COCOMO 81 모델

프로젝트 유형	공 식	유형 해설
유기형(Organic)	PM = 2.4 × (KDSI)$^{1.05}$	소규모 팀이 개발하는 잘 알려진 응용 시스템
반결합형(Semi-detached)	PM = 3.0 × (KDSI)$^{1.12}$	유기형과 임베디드의 중간형으로 트랜잭션 처리 시스템이나 운영체제, 데이터베이스 관리 시스템
내장형(Embedded)	PM=3.6 × (KDSI)$^{1.20}$	하드웨어가 포함된 실시간 시스템. 미사일 유도, 신호기 제어 시스템

COCOMO에 의한 비용 예측



기본 COCOMO 방법

< 예제1 > 32,000 LOC로 예상되는 Organic Mode

$E = 2.4*(32)^{1.05} = 91$ man-months

$D = 2.5*(91)^{0.38} = 14$ 개월

$N = 91 / 14 = 6.5 \fallingdotseq 7$ 명

-생산성

32,000 / 91 = 352 LOC/MM

352 / 22 ≒16

그러므로, 한 사람이 하루에 약16라인작성

기본 COCOMO 방법

< 예제2> 128,000 LOC의 크기인 Embedded Mode

$E = 3.6*(128)^{1.20} = 1216$ man-months

$D = 2.5*(1216)^{0.32} = 24$ 개월

N = 1216 / 24 = 50.66 ≒51명

-생산성

128,000 / 1,216 = 105 LOC/MM

105 / 20 ≒4

그러므로 한 사람이 하루에 약 4라인작성

COCOMO 노력 승수

	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	

COCOMO-81

- 단점
  - 프로젝트의 초기 단계에서 Size 값을 예측하는 것이 어려움
  - 기본 예측 모델에서 B와 M의 값에 영향을 주는 요소들이 주관적
  - 실제 대부분의 대형시스템은 서로 상이한 서브시스템으로 구성되며 이중 일부분은 Organic Mode이고 다른 부분은 Embedded Mode인 경우도 있다.
  - 보정(calibration)

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

추정 과정

1. 어플리케이션을 구성하는 화면, 보고서, 3세대 언어 컴포넌트의 숫자를 카운트
2. 화면과 보고서의 복잡도 수준을 결정
3. 화면과 보고서, 3세대 언어 컴포넌트를 위한 복잡도 가중치를 찾음
4. 화면, 보고서, 3세대 언어 컴포넌트의 개수에 가중치를 곱하여 객체 점수(Object Point)를 계산
5. 재사용률(reuse)을 예측하여 공식에 대입하여 NOP(New Object Point)를 구함
6. 객체 점수 생산성(PROD)을 결정
7. 객체 점수 생산성을 식 PM = NOP/PROD 에 대입하여 최종 PM(Person Month)값을 구함

COCOMO II 세가지 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 세가지 단계

비교대상	단계 1: 응용합성 (프로토타이핑)	단계 2: 초기 설계	단계 3: 설계 이후
프로덕트 비용 승수	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 비용 승수	없음	개발 기간, 개발 환경에 대한 요구	소프트웨어 도구 사용, 개발 기간, 여러 사이트 개발 요구

기능 점수

- 기능 점수(function points)
  - 정확한 라인수는 예측 불가능
  - 입력, 출력, 질의, 화일, 인터페이스의 개수로 소프트웨어의 규모를 나타냄
  - 각 기능에 가중값
  - 기능 점수 1을 구현하기 위한 LOC
    - 어셈블리 언어(324), C언어(150), Pascal(91), Ada(71), APL(32)
- 복합 가중값을 이용한 기능점수 산출
- 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.

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

(필기 참고)

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 점수 구하는 방법

- 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

기능 점수 구하는 방법

- [표 3.5]에 대입하여 GFP를 구한다.
  - GFP = 10 x 3 + 5 x 4 + 8 x 3 + 30 x 7 + 4 x 5 = 304 FP
- 처리 복잡도 보정 계수 구하면
  - PCA = 0.65 + 0.01(12 x 3 + 2 x 5) = 1.11
- FP를 보정
  - FP = GFP x PCA = 304 x 1.11 = 337.44 FP
- 추정 노력(E) = FP / 생산성 = 337.44 / 60 = 5.624 persons-week

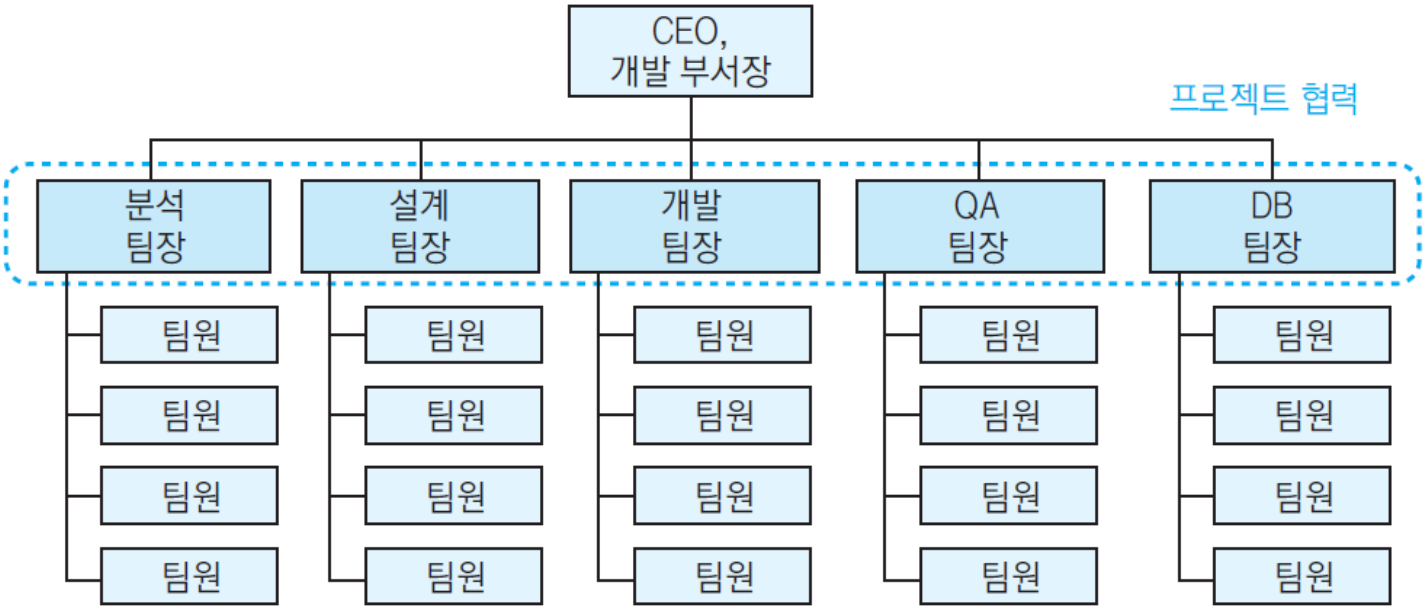
국내 기능 점수 산정 가이드

- 정보통신산업진흥원의 소프트웨어 공학에서 산정 기준을 제시[SW 산업본부, 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)

3.4 프로젝트 팀 조직

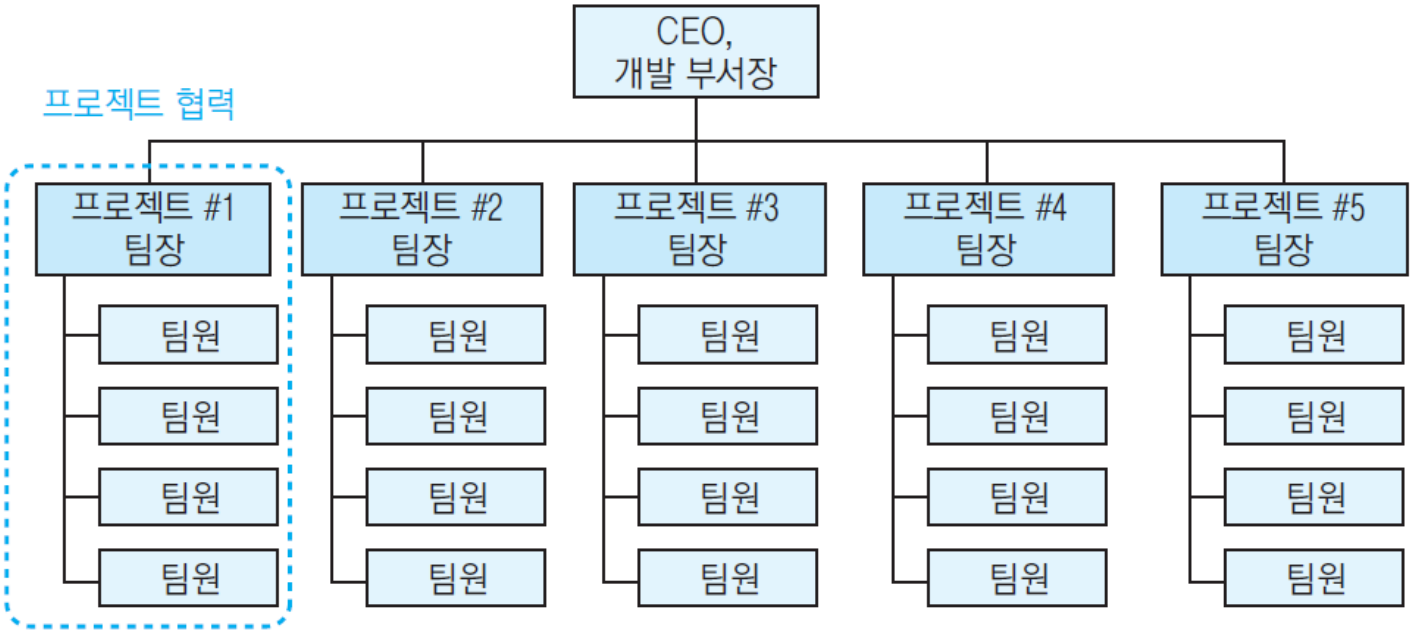
- 조직의 구성
  - 소프트웨어 개발 생산성에 큰 영향
  - 작업의 특성과 팀 구성원 사이의 의사교류
- 프로젝트 팀 조직 정의
  - 역할과 책임이 어디에 있는가?
  - 어떤 통로로 정보가 전달되고 결정되는가?
  - 어떻게 갈등을 해소할 것인가?
- 팀 역할 나누기
  - 프로젝트 관리자(project manager), 시스템 운영자(system administrator), 시스템 분석가(system analyst), 시스템 개발자(software engineer), 데이터베이스 엔지니어(database engineer), QA 관리자(QA manager), 기술 지원(technical support), 하드웨어 엔지니어(hardware engineer), 웹 개발자 및 디자이너

직능별 조직



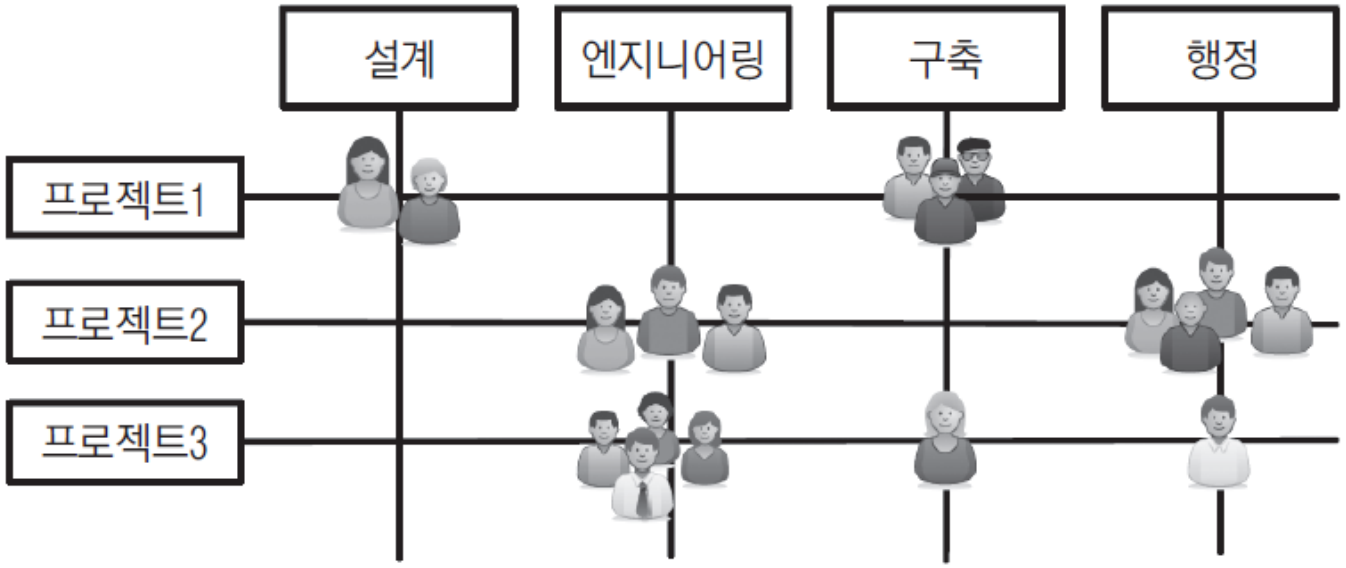
- 서로 다른 부서가 한 프로젝트의 다른 단계에 들어와 작업을 수행
- 팀원은 한 부서에 소속, 프로젝트의 협력은 부서별로

프로젝트별 조직



- 직능별 개발자들이 프로젝트에 배정
- 의사 전달 경로가 짧으며 인력, 진도 등 프로젝트 관리가 수월

매트릭스 조직



- 직능별 조직의 관리자가 프로젝트 책임을 맡고 직능별 조직 부서에 소속된 개발자가 프로젝트에 참여
  - 강한 메트릭스
  - 약한 메트릭스

애자일 조직

- 서로 밀접하게 협력하는 5~9명의 팀
- 결과와 이슈에 대한 오너쉽을 공유

역동적인 팀을 운영할때 이용

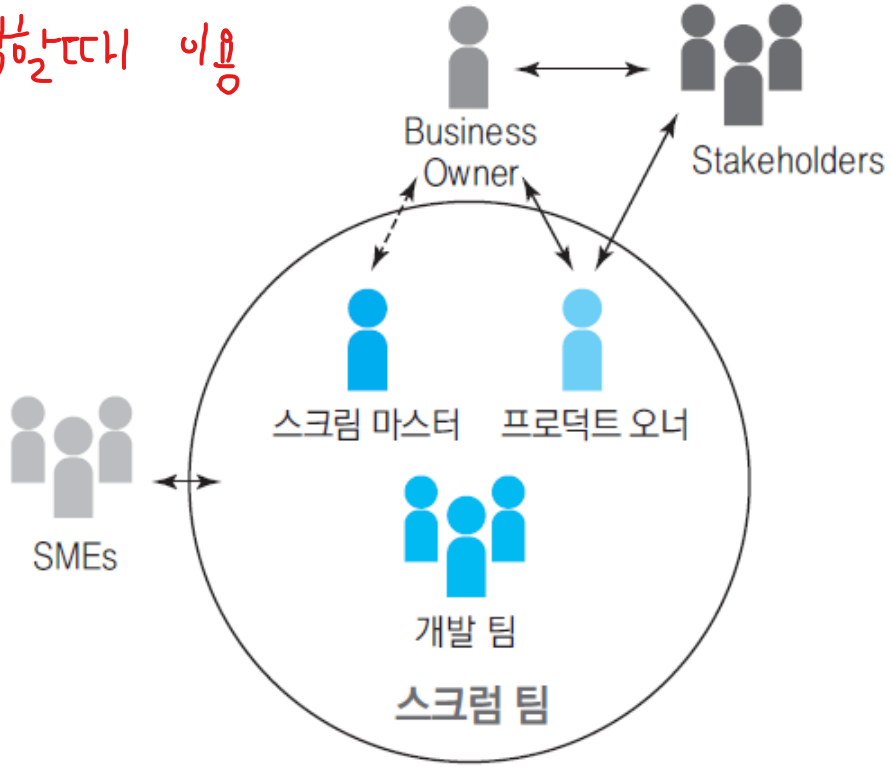


그림 3.15 애자일 조직

3.5 실행과 모니터링

- 프로젝트 실행
  - 작업 시작 미팅
  - 작업 결과 수집
- 프로젝트 모니터링

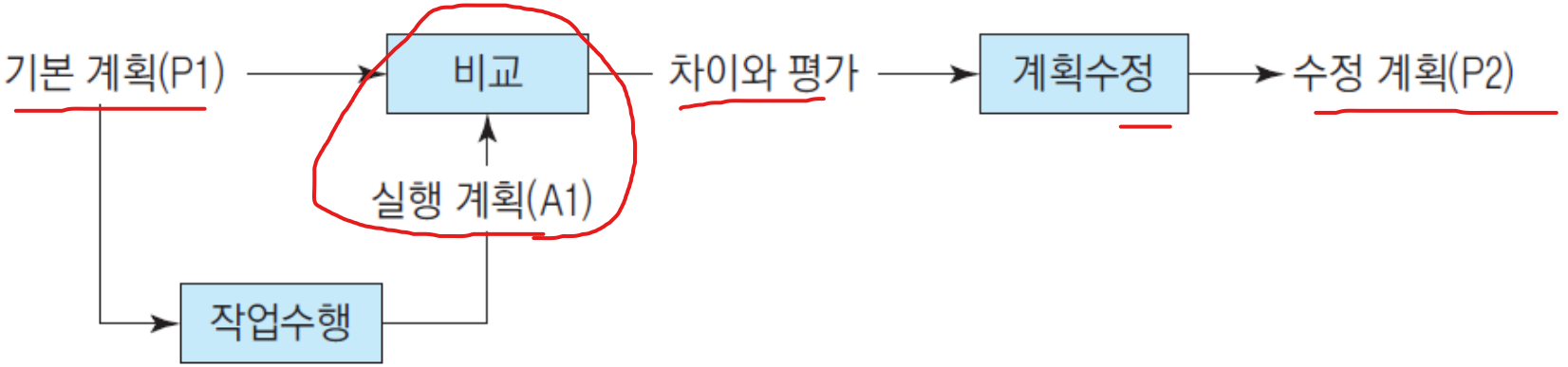
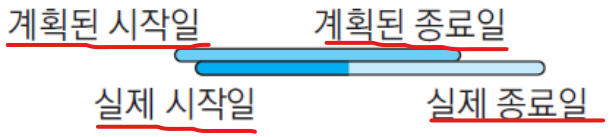


그림 3.16 모니터링

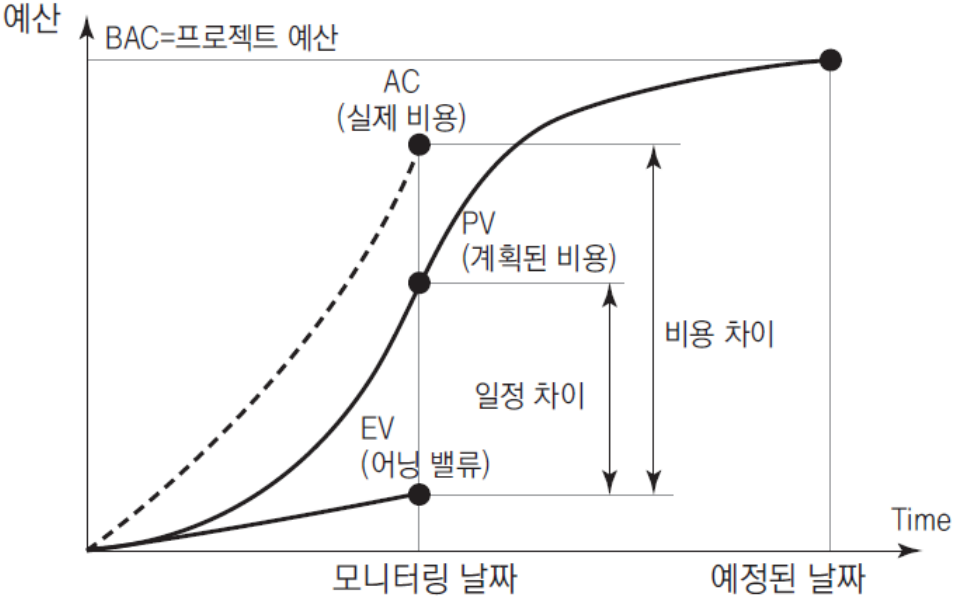
모니터링

- 일정 모니터링



홍길동	W1	W2	W3	W4
요구 M1	30		6	
요구 M2		30	6	
회의	2	2	6	2
연구	4	4	4	4
간접 활동				2

- 어닝 밸류 분석



3.6 리스크 관리

- 리스크 관리의 목적
  - 위험이 발생되었을 때의 영향을 줄이는 것

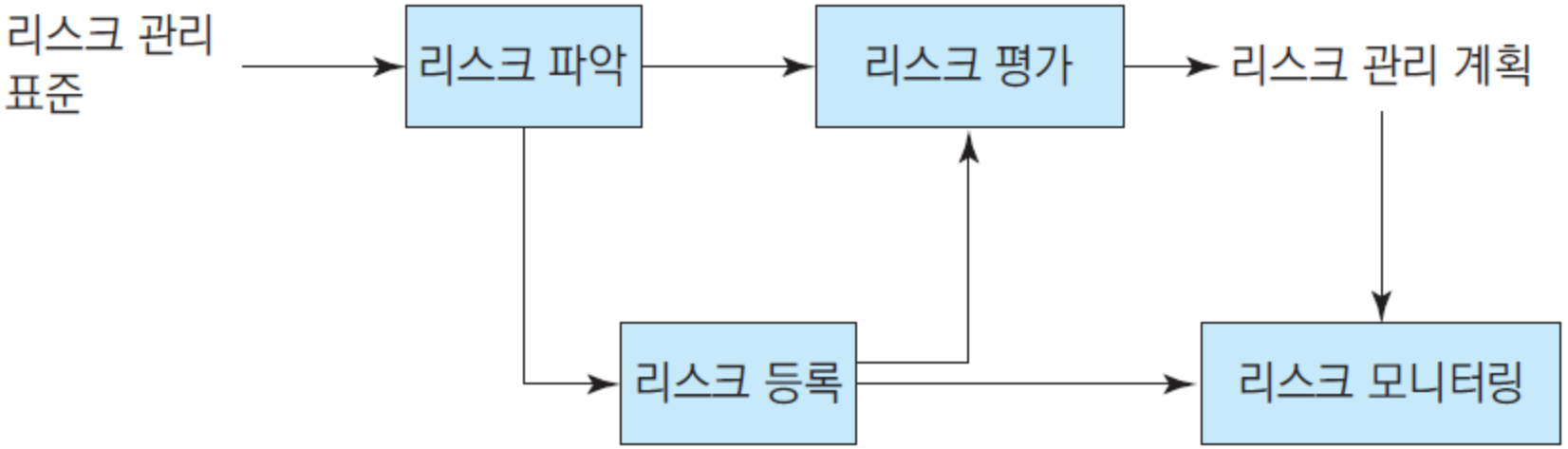


그림 3.20 리스크 관리

리스크 파악

- 리스크 찾는 방법
  - 회의
  - 문서 분석
  - 리스크 분할 구조, 체크리스트
  - 유추

6M(제조 관련 프로젝트)
- 기계(Machine)
- 방법(Method)
- 재료(Materials)
- 측정(Measurement)
- 사람(Man)
- 환경(Mother nature)

8P(경영, 서비스 관련 프로젝트)
- 가격(Price)
- 광고(Promotion)
- 인력(People)
- 프로세스(Process)
- 장소(Place/Plant)
- 정책(Policy)
- 절차(Procedure)
- 제품, 서비스(Product)

SW 관련 프로젝트(Boehm 제시)
- 인력부족
- 비현실적 일정 및 예산
- 잘못된 기능과 특징 개발
- 잘못된 인터페이스의 개발
- 과포장
- 계속적인 요구변경
- 외부에 보일만한 컴포넌트의 빈약
- 실시간 성능의 빈약
- 녹슨 컴퓨터 분야의 기술

그림 3.21 리스크 요소

리스크 평가

- 영향도에 따라 평가하고 우선순위를 매김
  - 확률과 리스크가 발생했을 때 미치는 영향이 우선 순위 좌우
- 정성적 방법
  - 확률을 모를 때

구분	경제적/환경적/사회적 결과				
리스크 발생 가능성	Negligible	Low	Medium	High	Extreme
Extremely high	H	H	E	E	E
High	M	H	H	E	E
Medium	L	M	H	E	E
Low	L	L	M	H	E
Negligible	L	L	M	H	H

그림 3.22 리스크 매트릭스

성장에는 시간이 걸린다.

- 호박과 토마토는 몇 주 만에 자라 며칠, 몇 주 동안 열매가 열리지만, 첫 서리가 내리면 이내 죽어버린다.
- 반면 나무는 서서히 몇 년, 몇 십 년, 몇 백 년까지 자라고 열매도 수 십 년 동안 맺는다.
- 건강하기만 하면 서리나 태풍, 가뭄에도 끄떡없다.

-존 맥스웰, '사람은 무엇으로 성장하는가'에서

- 빨리 자라면, 빨리 생을 마감하게 되는 것이 자연의 법칙입니다.
- 인생에서 중요한 일은 대개 예상보다 시간도 많이 걸리고 비용도 많이 듭니다.
- 일이 생각만큼 잘 안된다면 천천히 자랄수록 더 튼튼하게 자라는 거라 믿고 낙심하는 대신 기다릴 줄 아는 지혜가 필요합니다.