



쉽게 배우는 소프트웨어 공학

2판

Chapter 03 계획





목차

- 01 계획의 이해
- 02 문제 정의
- 03 타당성 분석
- 04 개발 비용 산정
- 05 비용 산정 기법 1: 하향식 산정 기법
- 06 비용 산정 기법 2: 상향식 산정 기법
- 07 비용 산정 기법 3: 수학적 산정 기법
- 08 일정 계획
- 09 위험 분석





학습목표

- 체계적인 계획을 세우는 방법을 알아본다.
- 소프트웨어 개발 비용 산정을 알아본다.
- 기능 점수 방법을 자세히 살펴본다.
- 프로젝트 일정 계획과 위험 분석 방법을 알아본다.



01. 계획의 이해



■ 계획

▪ 계획의 역할

- 계획을 제대로 세우지 않고 수행하는 소프트웨어 개발은 일정 지연, 비용 초과, 품질 저하라는 결과를 낳게 됨



그림 3-1 초등학생의 방학 중 하루 일과표와 개학 전의 모습



01. 계획의 이해



■ 계획

▪ 계획의 중요성

- 소프트웨어 개발의 성패는 비용, 기간, 인력과 같은 자원을 토대로 초기에 얼마나 계획을 잘 세우느냐에 달려있음



그림 3-2 프로젝트 마감일을 앞둔 프로젝트 팀의 모습



02. 문제 정의



■ 문제 정의

- 문제가 무엇인지 정의하는 것은 소프트웨어 개발의 첫단계임.
- 현재 상황과 구현될 시스템의 목표 및 제약 조건 등을 포함해 무엇을 개발할 것인지 명확히 정의하고, 개발 범위를 결정한다
- 프로젝트의 초기 타당성과 초기 계획을 작성할 수 있는 기초가 됨

■ 문제 정의

- 문제를 정의하려면 개발하고자 하는 영역의 배경 지식이 필요
- 유사한 프로젝트를 개발한 경험이 있는 분석가가 참여하는 것이 도움이 됨
- 문제를 파악하기 위해 현재 운영 중인 시스템을 사용해 봄
- 실무 담당자와 면담해 자료를 수집한 후 면밀히 분석해보는 것이 필요



03. 타당성 분석



■ 타당성 분석

▪ 경제적 타당성

- 경영자 입장에서 의사결정을 하는 데 매우 중요한 요소
- 시장 분석을 통해 시장성을 확인
- 경제적 타당성 분석으로 투자 효율성과 시장성을 검증한 후 개발 여부를 판단

▪ 기술적 타당성

- 사용자가 요구하는 프로젝트가 최신 기술이 필요하다면 기술적 타당성을 먼저 검토
- 요구하는 기술을 회사가 가지고 있는지 확인
- 부족하다면 필요한 기술을 갖고 있는 소프트웨어 개발자를 채용하거나 외주 개발로 해결

▪ 법적 타당성

- 오픈소스는 소프트웨어 개발에서 **분쟁** 발생 소지가 높음
- 소프트웨어 개발에서 오픈소스를 사용하는 것은 비용 절감 측면에서 매우 효율적
- 오픈소스는 원시 코드가 개방되어 있다는 것이지 아무렇게나 가져다 사용할 수 있는 것은 아님
- 법적인 문제가 발생하지 않으려면 오픈소스를 사용 할 때 어디까지 무료로 사용할 수 있는지 확인해야 함



04. 개발 비용 산정



■ 개발 비용 산정의 어려움

- 조립 컴퓨터 가격, 전자 제품 가격, 건축비 등은 어느 정도 근거가 명확해 소비자도 인정하기가 쉬움

부품	수량	부품 단가	합계	인건비	조립 컴퓨터 생산 원가
CPU	1개	232,000원			
메인보드	1개	244,000원			
RAM	2개	24,000원			
VGA	1개	277,500원			
SSD	1개	140,000원			
HDD	1개	39,800원			
ODD	1개	21,830원			
			979,130원	200,000원	1,179,130원

그림 3-3 조립 컴퓨터 생산 원가 산정의 예

- 소프트웨어 가격은 근거를 명확히 제시하기 어려우니 사용자도 받아들이기가 쉽지 않음
- 전자 제품의 경우 생산 공정이 꾸준히 개선되어 표준화 자동화된 방법이 있지만, 소프트웨어는 개발 프로세스가 다양하기 때문에 표준화나 자동화가 어려워 개발 프로세스에 따라 생산성이나 품질이 서로 다를 수 있음
- 개발자의 능력이 품질과 개발 기간에 영향을 미침



04. 개발 비용 산정



■ 개발 비용에 영향을 주는 요소

- 프로그래머 자질
 - 초급 프로그래머와 경험이 많은 프로그래머의 생산성에는 큰 차이가 있음
- 소프트웨어 복잡도
 - 브룩스의 법칙: “애플리케이션을 개발하는 것보다 유ти리티를 개발하는 것이 세 배 어렵고, 유ти리티를 개발하는 것보다 시스템 프로그램을 개발하는 것이 세 배 어렵다”
- 소프트웨어 크기
 - 개발하려는 소프트웨어의 규모가 크면 개발 인력과 개발 기간도 늘고 복잡도도 더 커짐
- 가용 시간
 - 관리자의 잘못된 생각 중 하나가 개발 기간을 단축하려면 인력과 자원을 늘리면 된다는 것. 그러나 인력을 많이 투입해도 개발 기간을 무한정 단축시킬 수 없음
 - 보엥: “정상적인 계획에서 최대 75%가 줄일 수 있는 한계”
- 요구되는 신뢰도 수준
 - 사고나 오류 발생 시 재산에 큰 손실을 끼치거나, 인명 피해가 발생할 수 있는 소프트웨어들은 개발 시 높은 신뢰도를 요구
- 기술 수준
 - 소프트웨어 개발 시 고급 언어를 사용하면 저수준 언어를 사용할 때보다 프로그래머의 생산성이 5~10배 높아짐
 - 개발 환경이 좋으면 그만큼 개발자가 할 일이 줄어들어 개발 기간을 단축할 수 있고 비용도 절감 가능



05. 비용 산정 기법 1: 하향식 산정 기법



■ 비용 산정 기법: 하향식

비과학적인 기법으로 조직 내 경험이 있는 2명 이상의 전문가에게 비용 산정을 의뢰하는 전문가 판단기법과 1명의 조정자와 다수의 전문가의 의견을 종합해 비용을 산정하는 델파이기법이 있음

▪ 전문가 판단 기법

- 경험이 많은 여러 전문가가 프로젝트를 수행하는 데 비용이 어느 정도 들어가는지 평가한 금액을 개발 비용으로 산정
- 경험이 많은 전문가가 판단을 내린 만큼 신뢰성이 있고 편리하다는 장점
- 짧은 시간에 개발비를 산정하거나 입찰에 응해야 하는 경우 많이 사용
- 수학적 계산에 의해 산정하지 않고 경험에만 의존할 경우 부정확할 수 있음



05. 비용 산정 기법 1: 하향식 산정 기법



■ 비용 산정 기법: 하향식

■ 델파이기법

- 전문가의 경험을 중요시해 비용을 산정하는 것은 같으나 전문가들의 편견이나 분위기에 영향을 받지 않도록 조정자를 둠
- 여러 전문가가 모여 각자의 의견대로 비용을 산정 후 결과를 공유하고 의견을 조율하여 개발 비용을 산정
 - ① 조정자는 전문가가 모여 비용 산정을 하는 회의에서 간사 역할을 수행
 - ② 전문가는 비용을 산정할 수 있는 자료를 충분히 검토하고, 필요하다면 의견을 나눌 수 있음
 - ③ 전문가 각자가 비용을 산정한다. 이때 계산된 결과를 조정자에게 익명으로 제출
 - ④ 조정자는 각 전문가가 제출한 자료를 요약 정리
 - ⑤ 조정자는 각 전문가가 제출한 자료에서 산정 내용에 차이가 크면 이 문제를 해결하기 위해 회의를 소집
 - ⑥ 전문가는 다시 익명으로 비용 산정 작업을 실시

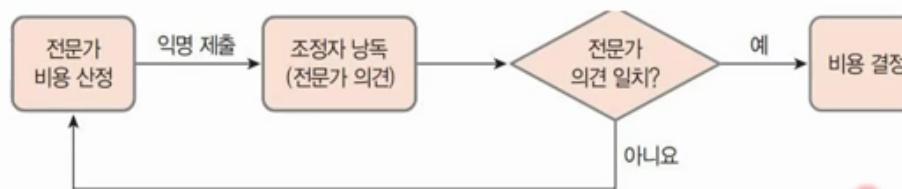


그림 3-4 델파이 기법의 비용 산정 방법



06. 비용 산정 기법 2: 상향식 산정 기법



- 상향식 산정 기법은 프로젝트의 세부 작업 단위별 비용을 산정한 후 전체 비용을 합산하는 방법
- 각 기능의 원시 코드 라인 수의 비관치(가장 많은 라인 수), 낙관치(가장 적은 라인 수), 중간치(기대치, 평균 라인 수)를 측정하여 예측치를 구해 비용을 산정하는 원시코드 라인 수 기법과 생명주기의 각 단계별로 노력[인/월수, M/M(Man/Month)]을 산정하는 개발 단계별 노력 기법이 있음
- 원시 코드 라인 수 기법

낙관치: 한 모듈의 라인 수를 가장 적게 생각할 때의 예상 라인 수(가중치 1 부여)

비관치: 한 모듈의 라인 수를 가장 많게 생각할 때의 예상 라인 수(가중치 1 부여)

중간치: 한 모듈의 라인 수를 보통이라고 생각할 때의 예상 라인 수(가중치 4 부여)

추정 LOC: $(\text{낙관치} + (4 * \text{중간치}) + \text{비관치}) / 6$

예 : 낙관치 300 LOC, 비관치 900 LOC, 중간치 600 LOC

$$\text{추정 LOC} : (300 + 4 * 600 + 900) / 6 = 600 \text{ LOC}$$



06. 비용 산정 기법 2: 상향식 산정 기법



■ 비용 산정 기법: 상향식

▪ 원시 코드 라인 수 기법

- 소프트웨어 각 기능의 원시 코드 라인 수 LOC의 비관치, 낙관치, 중간치를 측정해서 예측치를 구하고 이를 이용해 노력, 개발 비용, 개발 기간, 생산성 등의 비용을 산정하는 기법

$$\begin{aligned} \text{• 노력}(인/월수^{M/M}) &= (\text{참여 인원}/\text{월}) \times \text{개발 기간} = \text{추정 LOC}/1\text{인당 월평균 생산 코드 라인 수} \\ \text{• 개발 비용} &= (M/M) \times \text{단위 비용}(1\text{인당 월평균 인건비}) \\ \text{• 개발 기간} &= (M/M)/\text{참여 인원} \quad \text{• 생산성} = \text{LOC}/(M/M) \end{aligned}$$

• 예제

- 소프트웨어 개발 기간은 1년(12개월)이다. 5명의 개발자가 12개월 동안, 7명의 개발자가 5개월 동안 참여한다면 이 소프트웨어 개발의 노력 M/M은 얼마인가?
 - 풀이: $(5\text{명} \times 12\text{개월}) + (7\text{명} \times 5\text{개월}) = 60M/M + 35M/M = 95M/M$
- LOC 기법에 의해 예측한 총 라인이 50,000라인이고, 개발자가 10명 참여한다. 그리고 개발자들이 월평균 500라인을 코딩한다면 개발 기간은 얼마나 되는가?
 - 노력 $M/M = \text{원시 코드 라인 수} / (1\text{인당 월평균 생산 코드 라인 수}) = 50,000\text{라인}/500\text{라인} = 100M/M$
 - 개발 기간 = $(M/M)/\text{참여 인원} = 100(M/M)/10\text{명} = 10\text{개월}$

▪ 개발 단계별 노력 기법.

- 각 기능을 구현하는 데 필요한 M/M을 소프트웨어 개발 생명주기의 **작** 단계에 적용해 단계별로 산정



07. 비용 산정 기법 3: 수학적 산정 기법



- 수학적 산정 기법은 상향식 비용 산정 기법으로 개발할 소프트웨어의 규모 LOC를 예측한 후 소프트웨어의 종류에 따라 각 비용 산정 공식에 대입해 비용을 산정하는 COCOMO 방법
- 소프트웨어 생명주기의 전과정에 사용될 노력의 분포를 가정해 주는 Putnam 방법,
- 기능 점수를 구한 후 이를 이용해 비용을 산정하는 기능 점수 방법이 있음

■ COCOMO 방법

- 소프트웨어 개발 비용을 산정할 때 원시 코드의 크기, 즉 라인 수를 중심에 두는 방법. 즉 원시 코드의 라인 수가 많으면 그만큼 개발 비용이 많이 듬
- 미국 TRW사의 보엠이 1981년에 저술한 《소프트웨어 공학 경제학》에서 이 방법을 제안
- 먼저 완성될 소프트웨어의 크기(라인 수LOC)를 추정하고 이를 준비된 식에 대입해 개발에 필요한 M/M을 예측
- COCOMO 방법의 고려 사항
 - 프로그램 유형에 따른 가중치를 두어야 함
 - 개발하려는 소프트웨어를 제품, 컴퓨터, 개발자, 프로젝트의 4가지 특성에 따라 총 15가지로 분류한 후 인건비를 더 보정



07. 비용 산정 기법 3: 수학적 산정 기법



- **가중치 반영하기**

- 단순형 프로젝트

- 복잡도와 난이도가 비교적 높지 않은 업무용 소프트웨어가 이에 해당
 - 규모 정도이고, 크기는 50KDSI 이하

- 중간형 프로젝트

- 일반 업무용 소프트웨어보다 복잡하고 규모가 더 큰 운영체제, 데이터베이스 관리 프로그램 등이 이에 해당
 - 규모나 복잡도가 중간 정도이고, 크기는 300KDSI 이하

- 내장형(임베디드형) 프로젝트

- 자동화 기기나 전자 제품과 같은 하드웨어와 밀접하게 관련 있는 내장형 소프트웨어가 이에 해당
 - 크기는 300KDSI 이상



07. 비용 산정 기법 3: 수학적 산정 기법



- 수학적 산정 기법은 상향식 비용 산정 기법으로 개발할 소프트웨어의 규모 LOC를 예측한 후 소프트웨어의 종류에 따라 각 비용 산정 공식에 대입해 비용을 산정하는 COCOMO 방법 ←
- 소프트웨어 생명주기의 전과정에 사용될 노력의 분포를 가정해 주는 Putnam 방법,
- 기능 점수를 구한 후 이를 이용해 비용을 산정하는 기능 점수 방법이 있음 ←

■ COCOMO 방법

- 소프트웨어 개발 비용을 산정할 때 원시 코드의 크기, 즉 라인 수를 중심에 두는 방법. 즉 원시 코드의 라인 수가 많으면 그만큼 개발 비용이 많이 듬
- 미국 TRW사의 보엠이 1981년에 저술한 『소프트웨어 공학 경제학』에서 이 방법을 제안
- 먼저 완성될 소프트웨어의 크기(라인 수LOC)를 추정하고 이를 준비된 식에 대입해 개발에 필요한 M/M을 예측
- COCOMO 방법의 고려 사항
 - 프로그램 유형에 따른 가중치를 두어야 함
 - 개발하려는 소프트웨어를 제품, 컴퓨터, 개발자, 프로젝트의 4가지 특성에 따라 총 15가지로 분류한 후 인건비를 더 보정



07. 비용 산정 기법 3: 수학적 산정 기법



- **가중치 반영하기**

- 단순형 프로젝트

- 복잡도와 난이도가 비교적 높지 않은 업무용 소프트웨어가 이에 해당
 - 규모 정도이고, 크기는 50KDSI 이하

- 중간형 프로젝트

- 일반 업무용 소프트웨어보다 복잡하고 규모가 더 큰 운영체제, 데이터베이스 관리 프로그램 등이 이에 해당
 - 규모나 복잡도가 중간 정도이고, 크기는 300KDSI 이하

- 내장형(임베디드형) 프로젝트

- 자동화 기기나 전자 제품과 같은 하드웨어와 밀접하게 관련 있는 내장형 소프트웨어가 이에 해당
 - 크기는 300KDSI 이상



07. 비용 산정 기법 3: 수학적 산정 기법



■ COCOMO 방법

▪ 개발 인건비 초기 예측

- COCOMO 방법에서 제시하는 공식을 사용하면 개발 인건비^{M/M}의 초기 예측 값을 계산
- 규모가 같은 소프트웨어일 경우 기본형보다는 중간형에서, 중간형보다는 내장형에서 M/M이 더 많이 필요

표 3-1 투입 인력 산출 공식과 프로젝트 유형에 따른 상수 a, b값

$$PM = a \times (KDSI)^b$$

상수	유형별 값
a	단순형(2.4), 중간형(3.0), 내장형(3.6)
b	단순형(1.05), 중간형(1.12), 내장형(1.20)

표 3-2 프로젝트 유형에 따른 투입 인력 산출 공식

프로젝트 유형	공식
단순형	$PM = 2.4 \times (KDSI)^{1.05}$
중간형	$PM = 3.0 \times (KDSI)^{1.12}$
내장형	$PM = 3.6 \times (KDSI)^{1.20}$

- PM^{Person/Month}: 소프트웨어 개발에 필요한 인력(인원/월)
- KDSI^{Kilo Delivered Source Instruction}: 소프트웨어의 최종 원시 코드 라인 수

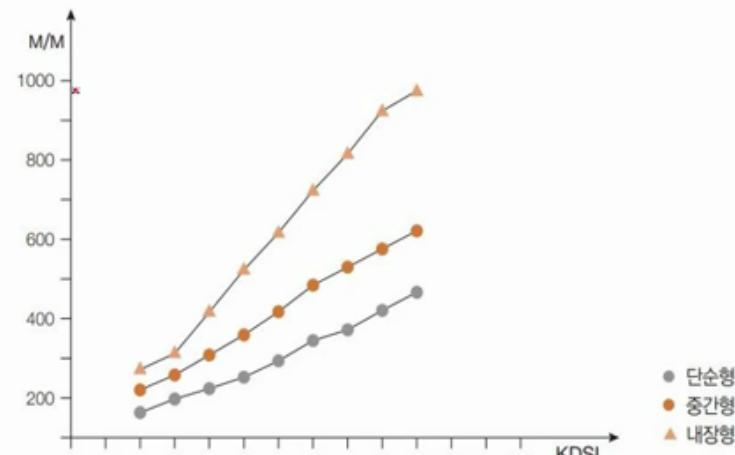


그림 3-5 COCOMO 방법에 의한 비용 예측



07. 비용 산정 기법 3: 수학적 산정 기법



■ COCOMO 방법

▪ 보정 계수 반영하기

- 노력 조정 수치(EAF): 보정에 사용하는 값, 필요한 각 항목별 승수 값을 모두 곱한 값
- 15개의 세부 항목에 따른 보정 값을 모두 정해 놓음
- 노력 조정 수치가 반영된 노력^{P/M}을 구하는 공식

표 3-3 COCOMO 방법에서 사용되는 노력 승수 값

특성	비용 승수 요소	승수 값					
		매우 낮음	낮음	정상	높음	매우 높음	극히 높음
제품 특성	요구되는 신뢰도	0.75	0.88	1.00	1.15	1.40	
	데이터베이스 크기	0.94	1.00	1.08	1.16		
	제품의 복잡도	0.70	0.85	1.00	1.15	1.30	1.65
컴퓨터 특성	실행 시간 제약			1.00	1.11	1.30	1.66
	주기억 장치의 제약			1.00	1.06	1.21	1.56
	HW/SW의 안정성		0.87	1.00	1.15	1.30	
	처리 시간		0.87	1.00	1.07	1.15	
개발자 특성	분석가의 능력	1.46	1.19	1.00	0.86	0.71	
	응용 분야 경험	1.29	1.13	1.00	0.91	0.82	
	컴퓨터와 친숙성	1.21	1.10	1.00	0.90	-	
	프로그래머 능력	1.42	1.17	1.00	0.86	0.70	
	프로그램 언어의 경험	1.14	1.07	1.00	0.95		
프로젝트 특성	소프트웨어 공학 기술 사용	1.24	1.10	1.00	0.91	0.82	
	소프트웨어 도구의 사용	1.24	1.10	1.00	0.91	0.83	
	요구되는 개발 일정	1.23	1.08	1.00	1.04	1.10	

표 3-4 노력 조정 수치가 반영된 유형별 노력^{P/M}

프로젝트 유형	공식
단순형	$PM=2.4 \times (KDSI)^{1.05} \times EAF$
중간형	$PM=3.0 \times (KDSI)^{1.12} \times EAF$
내장형	$PM=3.6 \times (KDSI)^{1.20} \times EAF$

(EAF : Effort Adjustment Factor)



07. 비용 산정 기법 3: 수학적 산정 기법



- 가중치 반영하기

- 단순형 프로젝트

- 복잡도와 난이도가 비교적 높지 않은 업무용 소프트웨어가 이에 해당
 - 중소 규모 정도이고, 크기는 50KDSI 이하

- 중간형 프로젝트

- 일반 업무용 소프트웨어보다 복잡하고 규모가 더 큰 운영체제, 데이터베이스 관리 프로그램 등이 이에 해당
 - 규모나 복잡도가 중간 정도이고, 크기는 300KDSI 이하

- 내장형(임베디드형) 프로젝트

- 자동화 기기나 전자 제품과 같은 하드웨어와 밀접하게 관련 있는 내장형 소프트웨어가 이에 해당
 - 크기는 300KDSI 이상



07. 비용 산정 기법 3: 수학적 산정 기법



- 수학적 산정 기법은 상향식 비용 산정 기법으로 개발할 소프트웨어의 규모 LOC를 예측한 후 소프트웨어의 종류에 따라 각 비용 산정 공식에 대입해 비용을 산정하는 COCOMO 방법 ←
- 소프트웨어 생명주기의 전과정에 사용될 노력의 분포를 가정해 주는 Putnam 방법,
- 기능 점수를 구한 후 이를 이용해 비용을 산정하는 기능 점수 방법이 있음 ←

■ COCOMO 방법

- 소프트웨어 개발 비용을 산정할 때 원시 코드의 크기, 즉 라인 수를 중심에 두는 방법. 즉 원시 코드의 라인 수가 많으면 그만큼 개발 비용이 많이 듬
- 미국 TRW사의 보엠이 1981년에 저술한 『소프트웨어 공학 경제학』에서 이 방법을 제안
- 먼저 완성될 소프트웨어의 크기(라인 수LOC)를 추정하고 이를 준비된 식에 대입해 개발에 필요한 M/M을 예측
- COCOMO 방법의 고려 사항
 - 프로그램 유형에 따른 가중치를 두어야 함
 - 개발하려는 소프트웨어를 제품, 컴퓨터, 개발자, 프로젝트의 4가지 특성에 따라 총 15가지로 분류한 후 인건비를 더 보정

