CHAPTER 03

프로젝트 계획과 관리

대규모의 복잡한 소프트웨어를 만드는 프로젝트에는 많은 인력이 참여한다. 소프트웨어가 높은 품질을 유지하며 효율적으로 개발이 진행되기 위하여 사전에 여러 가지를 예측하고 역할을 분담하는 계획을 하여야 한다. 또한 필요한 자원을 구성하고 개발을 진행시켜 상황을 파악하고 필요에 따라 수정하고 컨트롤하여야 한다. 이와 같이 소프트웨어 개발에는 적절한 관리 활동이 필요하다. 프로젝트 계획 단계에 필요한 비용, 일정의 견적 방법과 조직 구성, 모니터링을 위한 진척도 및 측정 방법도 소개한다.

contents

CHAPTER
08 프로젝트 계획과 관리

| 키포인트 |

- 프로젝트 시작단계에 해야 할 일은 무엇인가?
- 어떻게 목표를 정하고 일정을 계획하는가?
- 어떻게 프로젝트 비용을 산정하는가?
- 개발 팀은 어떻게 구성하는가?
- 프로젝트 수행을 어떻게 모니터링하고 제어하는가?
- 어떤 위험 요소가 있으며 이를 관리하는 방법은 무엇인가?

일반적으로 프로젝트는 정해진 기간 안에 한정된 자원으로 일정한 목적을 달성하기 위해 수행하는 업무를 말한다. 예를 들어 "새로운 상품이나 서비스를 개발하여 올해 1억 원의 매출을 올리겠다."는 목표를 세우고 활동을 시작했다면 프로젝트라 할 수 있다.

함께 상품이나 서비스를 개발하고 판매하는 일에 참여할 인원은 한정되어 있다. 또한 기간이 올해 안으로 정해져 있기 때문에 준비하는 시간도 한정된다. 이익을 올려야 하므로 재료와 비용에 대한 추정도 필요하다. 목표를 달성할 것인지 추적하고 결과를 판단할 수도 있어야 한다.

프로젝트 관리 목적은 작업 수행에 필요한 여러 가지 자원, 인력, 비용, 재료, 기술 등을 가장 효과적으로 사용하여 프로젝트의 목표를 달성하는 것이다. 프로젝트 관리는 소프트웨어 개발 이외에도 다른 영역의 업무에도 적용된다. 하지만 소프트웨어 개발 작업의 관리는 다음과 같은 측면에서 어려움이 따른다.

1) 개발 대상이 눈에 보이지 않는 소프트웨어이다. 건축, 토목 공사나 제품의 개발은 프로젝트의 진척도가 명확하다. 개발이 늦어지면 결과물이 보이지 않기 때문에 문제를 쉽게 파악할 수 있다. 하지만 소프트웨어는 관리자가 진도를 물체처럼 보거나 만져

서 파악하기 어렵고 개발 진행 상황을 판단하기 어렵다. 진척 상황도 개발자들이 작성한 문서나 원시코드, 또는 간접적인 측정 방법에 의존할 수밖에 없다.

2) 소프트웨어 분야는 조직마다 프로세스가 다르다. 건축이나 기계와 같이 오랜 경험에 의하여 표준화된 절차와 방법이 없다. 소프트웨어 개발 역사가 짧고 표준이 되는 개발 프로세스에 대한 지식과 경험이 충분하지 않다.

3) 소프트웨어 분야의 기술 발전은 매우 빠르다. 또한 프로젝트마다 특징이 다르다. 따라서 과거 프로젝트 진행에서 경험한 지식과 경험을 다음 프로젝트에서 적용하기가 어렵다. 프로젝트에 따라 발생하는 문제를 예측하고 대처하는 것이 쉽지 않다. 이런 특징으로 인하여 소프트웨어 프로젝트가 지연되고 품질이 저하될 가능성이 높다.

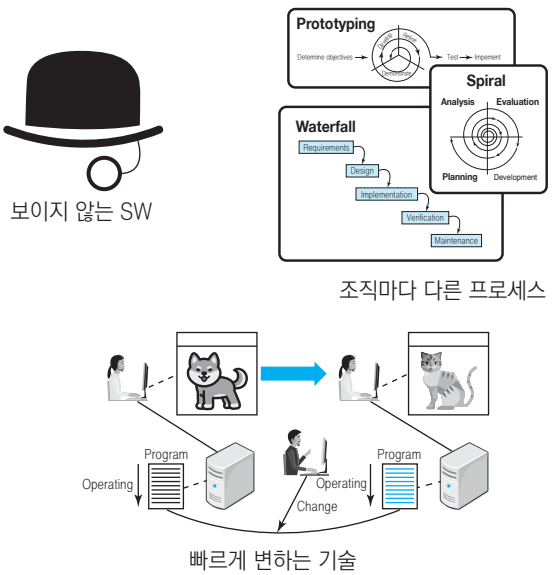


그림 3.1 SW 프로젝트 관리의 어려움

프로젝트 관리는 계획, 조직, 모니터링, 조정 등 네 가지 활동으로 이루어진다. 네 가지 활동은 서로 겹치는 부분도 있으나 프로젝트 진행에 따라 순차적으로 이루어진다. 프로젝트가 시작되면 진행 상황 정보를 수집하여 보고하며 필요에 따라 변경하는 조정 작업이 반복된다. 즉 모니터링과 조정이 프로젝트 관리의 핵심이다.

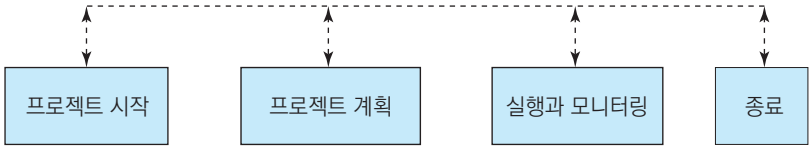


그림 3.2 프로젝트 관리 프로세스

프로젝트 관리자는 요구 사항을 파악하고 달성 가능한 여러 가지 목표, 기능, 품질, 범위, 시간, 비용 등을 달성하기 위하여 활동한다. 다음과 같은 목표가 프로젝트 관리의 중요한 목표이다.

- 최종 결과가 고객의 요구를 만족하여야 한다.
- 프로덕트 및 프로젝트에 대한 속성(품질, 보안, 생산성, 비용 등)이 요청 수준에 맞아야 한다.
- 계획된 일정에 맞게 진행되어야 한다.
- 팀 구성원이 효과적으로 작업하고 능력을 발휘하여야 한다.
- 실행 과정을 모니터링하고 조정하여야 한다.
- 프로젝트를 실패로 만들 수 있는 리스크를 예측하고 미리 대비한다.
- 요청된 도구와 기타 자원이 사용가능하고 효과적으로 쓰여야 한다.

이러한 목표로 프로젝트 관리 활동을 하지만 모든 것을 혼자 수행할 수는 없다. 프로젝트 팀 구성원이 소통하고 협력하여야 목표를 달성할 수 있다.

3.1 프로젝트 시작

프로젝트는 시장의 변화, 사회적 발전, 법이나 규정 개정, 기술적 발전 등의 필요에 의하여 기업이나 잠재적인 고객으로부터 시작된다. 프로젝트의 첫 작업은 목표를 세우고 가치와 리스크를 이해하는 일이다. 프로젝트를 시작할 것인가 말 것인가를 결정하는 요인은 다음 두 가지이다.

- 프로젝트가 제공할 가치 – 프로젝트에 의하여 창출되는 직접, 간접적 가치. 프로젝트 결과물이 얼마나 지속가능한지를 의미한다.

- 프로젝트와 관련된 리스크 – 자원의 가용성, 타이밍, 기술적 어려움과 불확정성 등

가치와 리스크는 많은 요소들과 연관되어 있어 평가하기는 쉽지 않으나 정량, 정성적으로 측정이 가능하다. 프로젝트 시작 단계에 올바른 가치 분석과 리스크 파악이 되어야 좋은 결과를 가져올 수 있다.

3.1.1 프로젝트 가치

프로젝트에 의하여 창출되는 직접, 간접 가치란 프로젝트의 긍정 및 부정적 결과를 측정하여 나타낼 수 있다. 가치는 매출이나 사회적 환경적 영향, 이미지 또는 지명도, 습득한 기술 등으로 측정할 수 있다. 또 다른 가치는 지속가능성(sustainability)이다. 프로젝트가 지속가능하고 종료 후 그 결과가 오래 동안 사용될 능력을 말한다. 하지만 가치는 매출과 개발된 시스템의 운영비용으로 나타낼 수 있기 때문에 지속가능성은 간과되는 경우가 많다.

프로젝트 가치를 평가하는 방법은 다음과 같은 방법이 있다.

1) 투자 회수 기간 – 투자금과 같은 금액을 벌어들이는 데 걸리는 기간.
2) ROI(Return of Investment) – 총비용에 대한 연간 평균 이익률
3) 순수 현재 가치 – 현재 투자금과 미래 수익금을 현재 가치로 비교하는 방법
4) 평가표 – 금액적인 요소 이외에 기술, 품질, 시간 여유, 인력 등을 고려하여 점수화하는 방법
5) SWOT – 프로젝트의 강점(Strength), 약점(Week), 기회 요인(Opportunity), 위험 요인(Threat)을 파악하여 타당성을 이해하는 방법

3.1.2 프로젝트 리스크

프로젝트는 일정한 기간 동안 인력, 재정, 기술적 자원이 필요하다. 자원을 미리 점유하기는 어려울 수도 있지만 프로젝트의 자원 요구를 미리 확인하는 것이 좋다. 고려해야 할 위험 요인은 자원, 현재 사용량과 가용성, 예상 사용량과 가용성, 프로젝트의 우선순위 및 중요도 등이다.

프로젝트의 결과는 정해진 기간 안에 마쳐야 하므로 시간이 또 다른 위험 요소가 될

수 있다. 프로젝트 결과를 너무 빨리 배포하거나 너무 늦게 하는 것은 의미가 없다. 유사한 소프트웨어를 경쟁사가 낼 수도 있기 때문이다.

기술적 어려움과 불확정성 또한 위험 요소이다. 프로젝트 성공의 열쇠는 여러 가지 기술적인 문제를 해결하는 능력에 좌우되기 때문이다. 기술적 문제가 무엇인지 이해하는 것 자체가 프로젝트와 관련된 리스크를 결정하는 중요한 요인이다.

3.1.3 타당성 분석

타당성 분석은 프로젝트를 공식적으로 인정하고 기관의 목표와 연결시키는 문서이다. 프로젝트의 타당성을 분석한 문서는 량이 많을 수도 있고 하나의 작은 프로젝트의 결과가 될 수도 있다. 관리자 입장에서는 프로젝트의 선택에 기초가 되는 문서다.

타당성 분석에 포함되어야 할 내용은 우선 목표와 제한 조건이다. 다음으로 위에서 설명한 프로젝트가 창출할 가치와 위험 요인이 기술되어야 한다. 중요한 것은 프로젝트의 목표가 품질, 비용, 시간 측면에서 달성 가능함을 보여야 한다.

1. SOW – 프로젝트가 성취하여야 할 일
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

그림 3.3 타당성 분석의 내용

3.2 프로젝트 계획과 스케줄링

프로젝트 계획 활동은 어떤 프로젝트든 첫 번째로 하는 작업이며 프로젝트의 성패는 계획의 치밀함에 달려 있다. 일정과 비용의 제한으로 계획 단계를 소홀히 하는 경우가 많다. 잘 계획된 프로젝트라도 변경과 재작업이 많기 때문에 치밀하고 실천 가능한 계획이 필요하다.

초기 계획 단계에는 다음과 같은 세 가지 사항에 대하여 계획한다.

- 목표 설정 – 프로젝트의 특성은 무엇이며 누가 자원을 제공하며 누가 사용할 것인지 정한다.
- 일정 정의 – 프로젝트 작업의 진행 스케줄과 할당할 자원을 정한다.
- 비용 추정 – 프로젝트를 완성시키기 위하여 필요한 비용을 추정한다.

계획 단계의 대부분의 일은 소프트웨어 프로젝트 관리자가 하여야 할 일들이다. 문제의 범위를 정하고 목표를 정하는 일, 일정을 정하는 일, 예산 및 자원 요구를 예측하는 일 등이다. 프로젝트에서 어렵고도 중요한 부분이 프로젝트의 범위와 필요한 작업, 일정을 예측하고 계획하는 일이다.

일정을 계획하는 순서는 그림 3.4와 같다. 우선 프로젝트의 목표와 요구를 정하고 문제의 범위를 정한다. 프로젝트의 문제를 정의한 문서(SOW)는 계획과 프로젝트 관리에 매우 중요하다. 다음은 문제를 작업 스케줄 관점으로 이해하기 위하여 WBS(Work Breakdown Structure) 분석이 필요하다. 작업의 의존관계를 파악하고 자원 할당을 고려하여 일정을 정하게 된다.

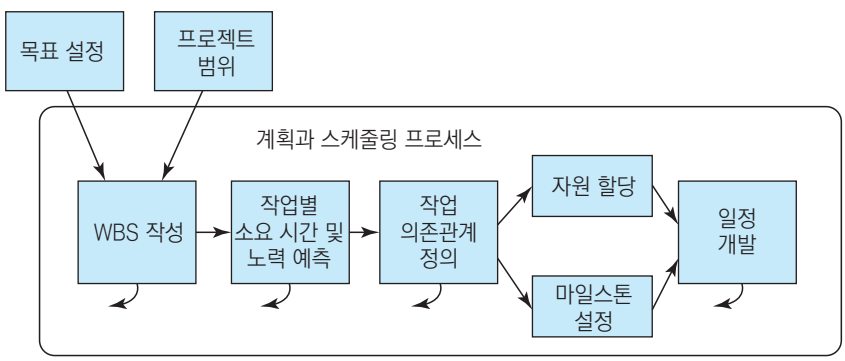


그림 3.4 프로젝트 일정 계획

3.2.1 목표 설정

프로젝트 목표를 정의하는 것은 범위를 정하는 일로 프로젝트 관리의 중요한 활동 중의 하나다. 프로젝트 목표에는 프로젝트에서 다루어야 할 모든 작업이 포함되어야 하며 범위 밖의 작업이 포함되어서는 안된다.

먼저 프로젝트에서 수행 할 필요한 작업의 기준을 설정한다. 또한 프로젝트 승인을 위

한 참조 문서를 정의합니다. 프로젝트 목표를 정하고 범위를 수정하는 작업은 타당성 조사부터 시작되어 프로젝트 진행되는 동안 계속된다.

- 프로젝트 범위를 문서로 정리한다면 다음 세 가지 내용이 포함되어야 한다.
- 프로젝트 목표 및 요구 – 프로젝트에서 달성하여야 할 목표와 수행되어야 할 기본 작업(WBS), 산출하여야 할 결과물, 승인 조건들이다.
- 가정과 제약조건 – 프로젝트가 성공하기 위하여 만족하여야 할 조건이다. 주로 프로젝트를 발주하는 입장에서 만족시켜 줄 것으로 기대하는 조건을 의미한다.
- 산출물과 점검 일정 – 프로젝트에서 정해진 일정에 산출하여야 할 결과물을 말한다.

3.2.2 프로젝트 범위

소프트웨어 개발 프로젝트를 위한 계획은 대상 업무나 문제의 범위(scope)를 정하는 것으로부터 시작한다. 사용자가 이해하는 용어로 정확히 범위를 정하여야 한다. 문제의 범위를 정하기 위해서는 현재의 상황을 잘 파악하고 구현될 시스템의 목표 및 제약조건 등을 정의하여야 한다.

문제의 범위는 사용자의 입장에서 작성하는 것이 중요하다. 즉, 문제를 기술할 때 특정 알고리즘이나 데이터베이스 모형, 하드웨어와 관련된 전문 용어를 사용하는 것은 피해야 한다. 프로젝트의 범위는 개발할 소프트웨어의 기능, 성능, 제약조건, 인터페이스, 신뢰성 등 프로젝트의 타당성과 초기 계획을 작성할 수 있는지를 결정하는 근거가 된다.

문제의 범위를 정의하기 위하여 먼저 문제의 배경과 응용 분야를 잘 이해할 필요가 있다. 이러한 지식을 얻기 위하여 사용자와 면담하거나 현장을 관찰하고, 필요에 따라서는 실제 업무를 수행해 보기도 한다.

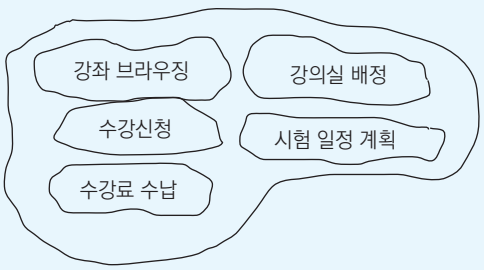
사례 ❶ 프로젝트 범위 정하기(수강 신청 시스템)

어떤 대학의 수강 신청을 위한 다음과 같은 소프트웨어를 개발한다고 할 때 프로젝트의 범위를 정해보자.

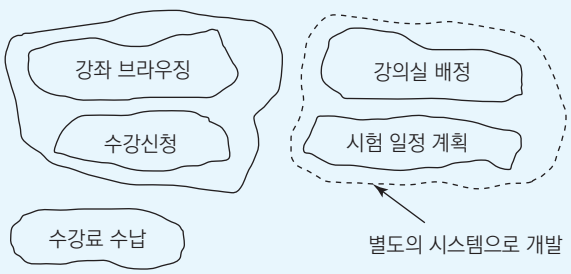
'시스템은 학생들이 쉽고 빠르게 강좌를 신청하고 변경할 수 있도록 하여야 한다. 또한 시스템은 학생들의 개인적인 목표, 즉 짧은 시간 내에 학위를 받으려는 목적과 흥미롭고 만족스러운 강의를 수강할 수 있도록 도와주어야 한다.'

문제가 광범위하고 부수적인 문제가 많으면 시스템은 범위가 크고 따라서 복잡할 것이다. 프로젝트 범위를 정하는 중요한 목적은 더 상세한 문제를 정의하여 범위를 좁히는 것이다. 예를 들어 위에서 설명한 수강 신청 시스템의 문제 정의에 '시스템은 교무행정실의 모든 기능을 자동화 한다'고 적었다면 매우 광범위하게 된다. 수강료 수납, 개설된 강의 리스트 작성, 강의실 지정 등의 모든 기능을 포함될 가능성이 열려 있기 때문이다.

범위를 정하는 방법은 시스템이 해결할 것으로 예상되는 문제들을 모두 리스트로 적는 것이다. 범위를 축소하려면 문제 리스트에서 몇 가지를 삭제하여 별도의 프로젝트로 남겨둔다. 그림 3.5에 있는 것이 범위 축소 과정을 나타낸 것이다.



(a) 광범위한 문제 범위



(b) 축소된 범위

그림 3.5 수강 신청 시스템 프로젝트 범위

3.2.3 WBS

프로젝트의 목표를 스케줄 관점으로 이해하기 위하여 WBS(Work Breakdown Structure) 분석이 필요하다. WBS는 개발 팀이 프로젝트 목표를 달성하고 결과물을 산출하기 위하여 수행하여야 할 작업을 계층적으로 분할한 것이다. 이러한 소작업들에 대한 소요 일정이 예측되어야 전체 프로젝트의 일정을 계획할 수 있다.

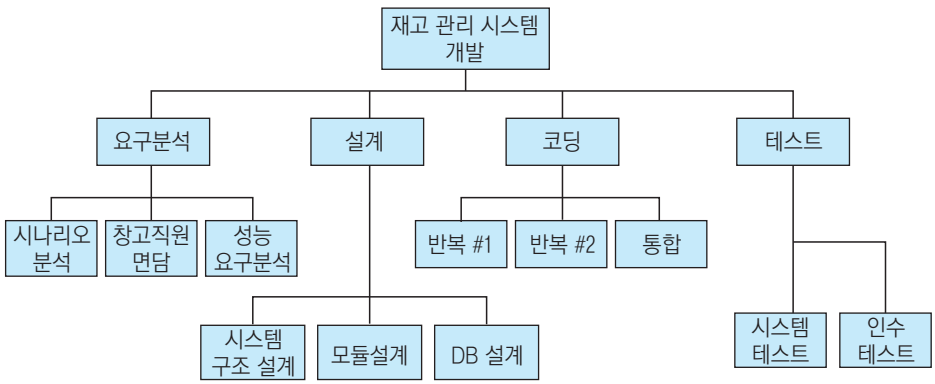


그림 3.6 재고 관리 시스템 개발을 위한 WBS

세부적으로 나누어진 WBS는 그림 3.6과 같이 나타낼 수 있다. '재고 관리 시스템' 개발 프로젝트의 주요 작업이 루트 아래 표시되어 있다. 트리의 모든 노드는 자식 노드에 표시된 소작업으로 나누어진다. WBS를 작성하는 목적은 프로젝트 진행에서 일어나는 모든 작업을 찾아내기 위한 것이다. 각 작업은 필요에 따라 더 자세한 소작업으로 나눌 수 있다. 예를 들면, '설계' 작업은 시스템 구조 설계, 모듈 설계, DB 설계로 이루어진다. WBS에서 최하위 단말 노드들의 수행에 필요한 자원의 예측은 프로젝트 전체 수행에 필요한 자원 예측에 기초 자료가 된다.

WBS는 그림 3.6과 같이 작업 프로세스 위주로 작성할 수도 있고 작업 단위를 산출물 위주로 작성할 수도 있다. 어떤 형식이든 최하위의 단위 작업은 프로젝트 단위 조직이 책임져야 할 규모의 목표가 되어야 한다.

WBS는 프로젝트에서 수행할 작업을 정의하는 데 효과적이며 목표와 관련된 작업을 잘 보여준다. 또한 작업과 비용을 정의하고 담당을 정하고 모니터링하는 데 유용하다. 또한 작업에 대하여 책임을 진 조직의 부서를 파악하는데 도움이 된다.

WBS는 프로젝트 스케줄링 작업의 입력이 된다. WBS를 통하여 프로젝트가 어떤 작업으로 이루어지는지를 알아낼 수 있고 스케줄링은 이들 작업을 어떤 순서로 할 것인가를 정하는 작업이다. WBS의 각 노드에 표시된 항목은 프로젝트에서 수행해야 할 작업이다. 스케줄링은 빠른 기간 내에 프로젝트를 완성할 수 있도록 이러한 작업의 순서를 최적화하는 일이다.

3.2.4 스케줄링

프로젝트의 목표가 설정된 후 하여야 할 일은 WBS를 기초로 하여 일정을 정의하는 것이다. 일정을 파악하여야 프로젝트에 투입될 인력의 규모와 비용을 추정할 수 있기 때문이다.

스케줄링 결과는 간트 차트로 표현되는데 다음과 같은 순서의 작업이 필요하다.

1) 작업 사이의 의존 관계 파악
2) CPM 방법을 이용한 여유 시간 계산
3) 소요 자원의 할당

■ 작업 의존 관계

작업을 수행하는 순서는 논리적으로 정해져 있다. 예를 들어 집을 지을 때 기초 공사를 하기 전에 지붕을 올릴 수는 없다.

작업 사이의 의존관계는 강한 관계와 약한 관계가 있다. 코딩한 후에야 테스팅 할 수 있다든지 계약이 끝나야 라이브러리를 사용하여 코딩을 시작할 수 있는 것이 강한 관계이다. 강한 의존관계를 제거하기 위해서는 재작업 비용이 든다.

약한 의존관계는 순서에 따라 다른 계획이 가능하다. 예를 들어 어떤 모듈을 먼저 개발할 것인지 순서를 정하는 것은 약한 의존관계이다. 약한 의존관계는 사용할 수 있는 자원과 수준에 따라 다른 순서로 진행할 수 있다.

작업 의존 관계와 소요 시간을 표로 나타내면 표 3.1과 같이 나타낼 수 있다. 선행 작업이란 어떤 작업을 시작하기 위하여 미리 완료되어야 하는 작업을 말한다. 예를 들면 C 작업의 선행 작업이 A로 표시 된 것은 C 작업이 시작되기 전에 A 작업이 끝나야 한다는 것을 의미한다. 즉, C가 소프트웨어 구조 설계 작업이라 한다면 선행 작업인 요구

분석 A 작업이 반드시 끝나 있어야 한다. 이러한 관계는 WBS 자체에는 나타나지 않는다.

표 3.1 작업 의존 관계

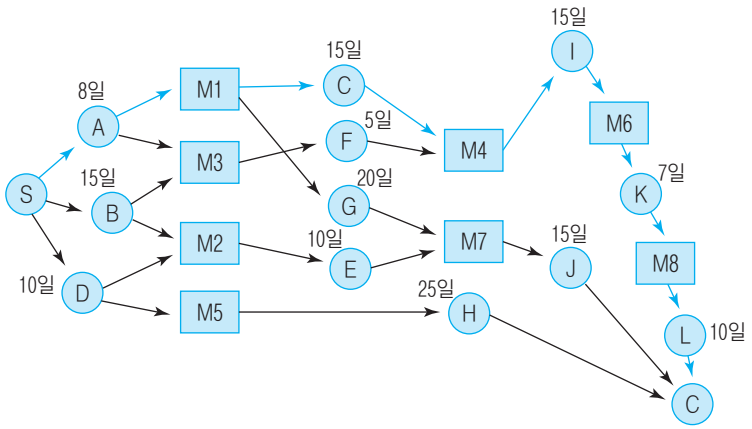
작업	선행 작업	소요 기간(일)
A	–	8
B	–	15
C	A	15
D	–	10
E	B, D	10
F	A, B	5
G	A	20
H	D	25
I	C, F	15
J	G, E	15
K	I	7
L	K	10

■ CPM 네트워크

CPM(Critical Path Method) 네트워크는 노드와 간선으로 구성된 네트워크이다. CPM 네트워크는 여러 가지 변형이 있다. 노드에는 작업을 표시하고 간선은 작업 사이의 선후 의존 관계를 나타낸다. 간선을 나타내는 화살표의 머리에 있는 작업은 화살표의 꼬리에 있는 작업이 끝날 때까지는 시작될 수 없다.

WBS에서와 같이 CPM 네트워크의 박스에는 작업의 시작일과 완성일을 표시한다. 이것으로 각 작업에 대한 가장 이른 시작일(Earliest Starting Time: TE)을 구할 수 있다. 일부 노드는 이정표(milestones)로 지정할 수 있다. 이정표는 프로젝트의 중요한 중간 결과를 완성하였다는 표시이다. 만일 이정표의 일을 완성시키지 못하였다면 일정을 수정하여야 한다는 것을 의미한다.

CPM 네트워크는 프로젝트 완성에 필요한 작업을 나열하고 작업에 필요한 소요 기간을 예측하는 데 사용한다. 이를 위하여 각 작업의 선후 관계를 결정하여야 한다. CPM 네트워크는 어떤 작업이 필요하고 각각이 얼마나 걸리며 각 작업의 선후관계를 한눈에 볼 수 있도록 나타내어 전체 프로젝트의 최소 소요 기간을 구하는 데 사용한다.



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

그림 3.7 CPM 네트워크와 임계 경로

■ 임계 경로와 여유 시간 계산

그림 3.7의 예에서 CPM 네트워크를 분석하면 S-A-M1-C-M4-I-M6-K-M8-L-X 작업으로 이루어진 경로가 소요 기간(55일)이 가장 긴 경로, 즉 임계 경로(critical path)가 된다. 임계 경로는 이 경로 상에 있는 어떤 작업이라도 늦추어지면 전체 프로젝트가 지연된다는 것을 의미한다. 관리자는 다른 작업보다 이런 작업들에 보다 관심을 두고 점검하여야 한다.

다음은 CPM 네트워크를 이용하여 각 작업의 여유 시간(slack time)을 구한다. 여유 시간은 다음의 식에 의하여 구할 수 있다.

$$TS = TL(\text{Latest Start Time}) - TE(\text{Earliest Start Time})$$

TE는 그 작업을 시작할 수 있는 최단 시간이다. CPM 네트워크를 이용하여 각 작업이 최대한 빠르게 시작할 수 있는 날짜, TE를 계산할 수 있다.

예를 들어 작업 C의 최대로 빠른 착수 일, TE는 8일이 지난 다음 날이다. 여기서 15일 지난 후, 즉 23일이 경과하여야 작업 C를 끝낼 수 있다. 한편 작업 C를 최대로 늦추어 시작할 수 있는 시간, TL은 최대경로(55일)에서 C(15일), I(15일), K(7일), L(10일)의 총 경과 기간을 빼면 8일이다. 최대로 빠른 착수일(TE)와 최대로 늦추어 시작할 수 있는 날(TL)이 같으므로 여유 기간이 없다. 따라서 C 작업은 임계경로에 있음을 알 수 있다.

른 작업 E의 최대 빠른 착수일(TS)과 늦은 착수일(TL)을 구해보자. 작업 E는 B가 끝나는 15일 이후에야 착수할 수 있다. 늦은 착수일(TL)은 55일에서 E(10일), J(15일)의 경과 기간을 뺀 30일이며, 즉 15일의 여유 기간(slack time)이 있다. E 작업은 임계경로에 있지 않기 때문이다.

■ 자원 할당과 간트 차트

각 작업에 대한 여유 시간을 구한 후 작업 별로 시작과 종료 기간을 수평 막대로 표현한 것이 그림 3.8과 같은 간트 차트이다. 프로젝트 관리 도구를 이용하면 간트 차트를 자동으로 그릴 수 있다. 여기서 작업 2는 3주의 여유 시간이 있다는 것을 알 수 있다.

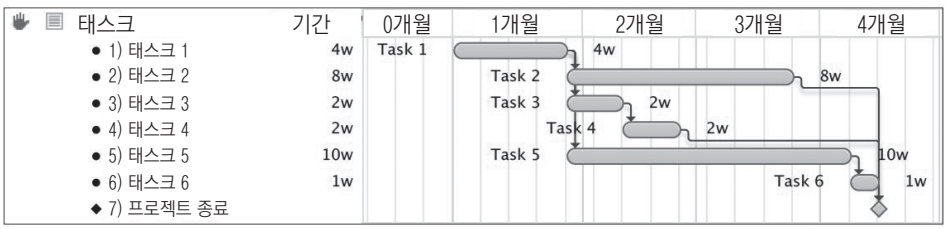
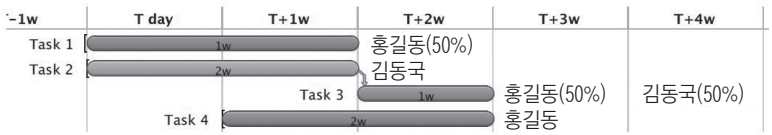


그림 3.8 간트 차트

프로젝트 진행을 위하여 일정 계획에는 필요한 자원을 할당한다. 소프트웨어 개발에 필요한 자원은 크게 세 가지로 나눌 수 있다.

- 인력 – 주어진 작업을 수행할 인원과 투입율
- 장비 – 주어진 작업을 수행할 때 필요한 도구나 하드웨어 및 소프트웨어
- 재료 – 주어진 작업을 수행할 때 필요한 소모품이나 자료

인력은 간트 차트에 표시된 작업 단위로 할당한다. 예를 들어 그림 3.8의 간트 차트에서는 작업 1에 개발자 홍길동을 50% 투입하고 작업 2에는 개발자 김동국을 100% 투입하였다. 소요 자원은 인력이 작업에 투입된 시간으로 나타낸다. 간트 차트의 각 열에는 투입된 자원의 총량이 표시된다.



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

애자일 프로세스의 스케줄링

CPM 네트워크와 간트 차트를 사용하는 전통적인 프로세스의 일정 계획과는 달리 애자일 프로세스는 스토리 카드와 같은 도구를 사용한다. 스토리 카드에는 사용자 스토리, 비즈니스 우선순위, 스토리 점수, 연관된 스토리 등이 적혀있다. 스토리 카드는 단어 암기 카드나 포스트잇 같은 것을 이용하면 우선순위나 연관성 등을 기준으로 쉽게 정렬할 수 있다.

애자일 프로세스의 일정 계획은 [그림 3.10]과 같이 카드나 포스트잇으로 빠르고 쉽게 만들 수 있다. 사용자 스토리를 적은 카드를 반복 단계별로 정리하여 수직으로 나열한다. 카드에 적힌 스토리 점수는 유스케이스를 개발하기 위하여 소요되는 노력을 의미한다. 스케줄링을 위하여 애자일 계획에서도 생산성 데이터가 필요하다. 이 팀은 스토리 점수 3점을 개발하려면 1주가 소요되었다.

그림 3.10 애자일 프로세스의 스케줄링

애자일 계획은 다음과 같은 장점이 있다.

- 높은 우선순위를 가진 유스케이스가 조기에 개발되어 설치된다는 확신을 줄 수 있다.
- 유스케이스 사이에 선행 관계를 지킬 수 있다. [그림 3.10]에서 UC3은 반복 주기 #2에 할당된 이유는 UC1과 UC9가 선행되어야 하기 때문이다.
- 각 열의 점수의 합은 개발 팀의 작업 속도를 초과하지 않아야 한다. 작업 속도는 생산성 통계값으로 팀이 반복 주기를 완성하는 데 드는 평균 노력이다.

3.3 비용 예측 기법

프로젝트 계획 작업에서는 소프트웨어를 개발하기 위하여 드는 비용이 얼마이며 어느 정도의 기간이 걸리는지를 예측할 필요가 있다. 이러한 예측은 개발이 시작되기 전에 이루어져야 하는데 그 이유는 비용 대 수익의 분석, 프로젝트 모니터링 및 관리에 필요하기 때문이다. 소프트웨어 개발을 위한 입찰에서는 비용의 예측이 아주 중요하므로 개발 노력에 대한 예측 기법을 알아야 한다.

비용을 계산하기 위해서는 우선 노력(effort)과 자원(resource)과 기간(duration)의 관계를 이해하여야 한다. 노력이란 프로젝트를 완성시키기 위하여 필요한 작업의 량이다. 자원은 작업에 동원될 수 있는 인력의 량이며 기간이란 작업이 수행될 기간이다.

노력은 작업이 완료되기 위하여 필요한 노동의 량이다. 단위는 일(Work-day), 주

(Work-week), 달(Work-month)로 표현할 수 있다. 예를 들어 세 사람이 일 개월 동안 작업한 노력은 3 MM(Man-month)라고 나타낸다.

노력과 자원, 기간의 관계는 다음 식으로 나타낼 수 있다.

$$D = E/M$$

여기에서 M(Manpower)은 투입되는 인력의 투입 비율의 총합이다. 예를 들어 80 시간의 노력에 대하여 2명을 100% 투입한다면 80시간/2명×40시간 = 1주의 기간이 소요된다. 만일 같은 노력에 대하여 1명의 인원이 50% 투입된다면 80시간/1명×20시간 = 4주의 기간이 필요한 것이다.

소프트웨어 개발 비용의 대부분은 소요되는 인력에 대한 비용이기 때문에 대부분의 비용 예측은 MM으로 표현하는 노력의 추정에 초점을 두고 있다. 즉 소프트웨어 공학에서의 비용 예측은 다른 엔지니어링 작업과는 크게 다르다. 건축이나 토목, 기계 등의 엔지니어링 작업에서의 비용은 대부분 벽돌, 알루미늄, 철강 등 원자재가 얼마나 사용되는가에 달려 있지만 소프트웨어 엔지니어링에서 비용 예측의 중요한 변수는 투입되는 엔지니어의 인원수와 작업 기간이다. 인건비에 하드웨어, 소프트웨어 등의 재료비와 사무실 임대료 등 오버헤드를 추가하면 개발 비용을 산출할 수 있다.

식은 간단하지만 프로젝트에 들어갈 노력을 추정하는 일은 쉽지 않다. 계획은 불확실하며 소프트웨어 프로젝트를 계획할 때는 대부분 낙관적인 시각으로 추정하기 때문이다. 불확정성을 계획에 포함시킬 수도 있다. 문제가 생겨 지연되었을 때를 고려하여 응급 시간을 미리 나타내는 것이다. 또 다른 방법은 지연이 발생하였을 때 계획을 바로 변경하는 방법이 있다.

비용 예측 기법에는 다음 세 가지가 있다.

- 전문가 판단 - WBS를 기초로 경험 많은 전문가가 각 작업에 소요되는 비용을 추정한다.
- PERT(Program Evaluation and Review Technique) - 기간을 결정하기 위하여 가중 평균, 즉 기대치를 계산하는 방법이다. 하나의 작업에 세 가지 관점의 추정치를 구한다. 낙관적(Optimistic)일 때 소요될 것으로 예측되는 기간(a), 보통(Most

likely) 때 소요되는 기간(b), 비관적(Pessimistic)일 때 소요될 것으로 예측되는 기간(c)이다. 기대치는 다음의 공식으로 구할 수 있다.

$$t_e = \frac{(a+4m+b)}{6}$$

- 알고리즘식 방법 – 프로젝트에 소요되는 노력을 체계적으로 결정하는 방법이다. 프로젝트의 측정 가능한 특징을 찾아 함수를 정의한다. 이미 개발이 종료된 프로젝트에 대한 데이터 $<a_1, ..., a_n, \text{effort}>$ 집합을 가지고 있다면 $f(a_1, ..., a_k) \propto \text{effort}$ 라는 수식에서 변수 사이의 관계를 찾을 수 있다.

알고리즘식 방법 중 널리 알려진 방법은 COCOMO, 기능 점수, 객체 점수 모델이다. COCOMO의 초기 모델과 COCOMO II, 기능 점수 방법에 대하여 알아본다.

3.3.1 COCOMO-81

B. Boehm이 제안한 원시 프로그램의 규모에 의한 방법이 1981년 COCOMO(Constructive Cost Model) 초기 모델이다. 먼저 완성될 시스템의 규모(Lines Of Code)를 추정하고 이를 준비된 식에 대입하여 소요 MM를 예측한다.

규모 기반 모델은 프로젝트 노력을 산정하기 위하여 규모를 기반으로 하는 수학적 공식을 사용한다. 여기에 개발 소프트웨어의 유형이나 팀, 프로젝트 프로세스, 등 프로덕트에 영향을 주는 요인을 고려한다. 즉 규모 기반 모델은 완성된 프로젝트의 비용과 속성을 분석하고 실제 경험에 근거한 예측 공식을 찾아서 구축할 수 있다.

소프트웨어 개발에 드는 노력을 규모 기반으로 추정하는 모델은 다음과 같은 기본 공식 형식을 따르고 있다.

$$\text{노력} = A \times (\text{Size})^B \times M$$

A는 개발 기관의 특징과 개발 소프트웨어의 유형에 좌우되는 상수이다. Size는 개발될 소프트웨어의 원시코드 라인 수나 기능 점수에 해당한다. B는 1에서 1.5 사이의 값으로 프로젝트에 대한 노력은 규모에 선형적으로 비례하지 않는다는 것을 의미한다. 소프

트웨어의 규모와 복잡도가 증가할수록 커뮤니케이션, 관리, 통합 등의 부담이 늘어나기 때문에 추가 비용이 발생한다. 시스템이 복잡할수록 이런 요인이 영향을 주는 지수 함수 형태이다.

초기 COCOMO-81 모델에서는 규모를 KDSI(Kilo Delivery Source Instruction)로 정하고 세 가지 프로젝트 유형에 따른 산출 공식을 [표 3.2]와 같이 예시하고 있다.

표 3.2 COCOMO 81 모델

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

프로젝트의 유형에 따라 예측한 라인수로 프로젝트 비용을 구하여 그 추이를 그래프로 나타내면 [그림 3.5]와 같다.

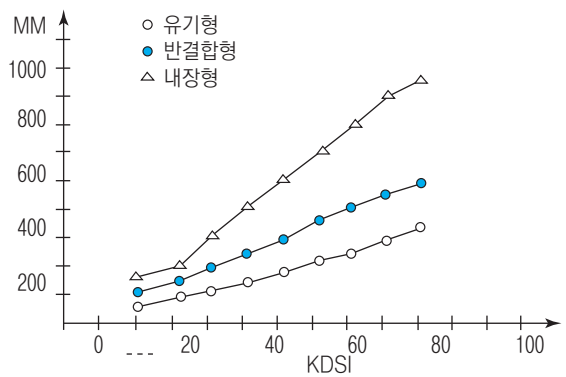


그림 3.11 COCOMO-81에 의한 비용 예측

예측 모델의 기본 공식에서 M은 프로젝트에 영향을 주는 여러 가지 다른 요소들, 예를 들어 제품의 성격, 목표 컴퓨터 시스템의 특성, 개발 구성원의 특성, 프로젝트 자체의 성격 등에 따라 달라지는 계수(multiplier)이다. M은 기본적으로 예측한 노력을 보정

(scaling)하기 위한 것이다. 예를 들어 같은 규모라도 프로그램의 경험이 많다면 노력 예측 값은 더 작아져야 하며, 실시간 처리 요구가 있다면 예측 값이 더 커져야 한다. 정확한 노력 예측을 위하여 여러 가지 비용 승수(cost-driver) 요소가 결정되어야 한다. Boehm이 제시한 비용 승수 요소는 [표 3.3]에 나열된 것들이며 각 승수의 값을 기본 노력 예측 값에 곱하면 된다.

표 3.3 COCOMO-81 모델에서 사용되는 노력 승수값

비용 승수 요소	승수값					
	매우낮음	낮음	정상	높음	매우높음	극히높음
제품의 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
컴퓨터의 특성						
실행 시간의 제약			1. 00	1. 11	1. 30	1. 66
주기억 장치의 제약			1. 00	1. 06	1. 21	1. 56
H/W, S/W의 안전성		0. 87	1. 00	1. 15	1. 30	
처리 시간		0. 87	1. 00	1. 07	1. 15	
개발 요원의 특성						
분석가의 능력	1. 46	1. 19	1. 00	0. 86	0. 71	
응용 경험	1. 29	1. 13	1. 00	0. 91	0. 82	
컴퓨터와의 친숙성	1.21	1.10	1.00	0.90	-	
프로그래머 능력	1. 42	1. 17	1. 00	0. 86	0. 70	
프로그래밍 언어 경험	1. 14	1. 07	1. 00	0. 95		
프로젝트 성격						
소프트웨어 공학 원리의 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	

■ 단점

규모를 바탕으로 제안된 대부분의 비용 예측 방법과 같이 COCOMO-81은 다음 문제점을 가지고 있다.

1. 프로젝트의 초기 단계에서 Size 값을 예측하는 것이 어렵다. 프로젝트 초기 단계에는 요구 명세 밖에 없기 때문이다. 초기 단계에는 원시코드 라인 수의 예측보다는 기능의 규모를 예측하는 것이 더 쉽다.

2. 기본 예측 모델에서 B와 M의 값에 영향을 주는 요소들이 주관적이다. 사람마다 기관마다 개발 될 시스템에 대한 경험 수준에 따라 달라진다.

최종 프로그램의 크기는 설계 단계의 의사결정에 의하여 크게 좌우되는데 초기 단계의 크기 예측은 이런 의사 결정이 이루어지기 이전이다. 예를 들어 고성능의 데이터 관리가 요구되는 시스템을 구현할 때 자체 데이터 관리 시스템을 이용할 수도 있고 상용 데이터베이스 시스템을 사용할 수도 있다. 초기 단계에 규모를 예측할 때는 고성능 상용 데이터베이스 시스템이 있는지 알 수 없고 따라서 데이터 관리 코드가 얼마나 포함될지 알 수가 없다.

소프트웨어 개발에 사용되는 프로그래밍 언어는 개발된 원시코드 크기에 영향을 준다. Java와 같은 언어를 사용하면 C 언어를 사용하는 것보다 라인 수가 좀 더 많아진다. Java는 컴파일 타임 체킹 등의 이유로 코드 크기가 늘어나는 데 이로 인하여 검증이 용이해질 수 있다. 또한 Java 언어는 이전 프로젝트의 원시코드가 재사용되는 경우가 많은데 크기 예측에 난감하게 될 수 있다.

규모 기반 모델의 또 다른 단점은 보정(calibration)이다. 규모 기반 모델을 사용할 때는 과거 프로젝트 데이터를 이용하여 모델과 속성값을 계속 보정해 주어야 한다. 하지만 모델을 수정할 만큼 충분한 데이터를 수집하는 기관이 드물다. 따라서 COCOMO-81과 같이 공개된 속성값을 사용하게 되는데 이는 프로젝트 팀 고유 환경과 얼마나 일치하는지 파악하기가 어렵다.

3.3.2 COCOMO II

1995년에 발표된 COCOMO II에서는 소프트웨어 개발 프로젝트가 진행된 정도에 따라 세 가지 다른 모델을 제시하고 있다. 개발 초기 단계에 LOC를 정확히 예측한다는 것은 어려운 일이므로 단계에 따라 다른 값을 예측하고 이를 바탕으로 소요되는 노력을 추정한다. 1단계는 프로토타입을 만드는 단계로 화면이나 출력 등 사용자 인터페이스, 3세대 언어 컴포넌트 개수를 세어 응용 점수(application points)를 계산하고 이를 바탕으로 노력을 추정한다.

2단계는 초기 설계 단계로 대안이 되는 자세한 구조와 기능을 탐구할 수 있어 1 단계보다는 더욱 정확한 예측이 가능하다. 2단계에서는 소프트웨어의 규모 측정을 위하여

다음 절에 설명할 기능 점수 방법을 채택하고 있다.

3단계는 구조 설계 이후 단계(post architecture)로 시스템에 대한 자세한 이해를 바탕으로 COCOMO-81에서 제안된 LOC에 의하여 소요되는 노력을 추정하는 방법을 사용할 수 있다. COCOMO II에 적용되는 기본 모델은 다음과 같다.

$$E = b\, S^c\, m(X)$$

여기에서 기초 소요노력 예측 값이 b S^c이며 m(X)는 비용승수의 벡터이다. [표 3.4]은 각 단계에 적용되는 비용승수를 나타내었으며 규모 측정 방법 이외에 초판 COCOMO-81과 COCOMO II의 다른 점을 정리하였다. COCOMO II는 재사용, 요구 분석, 요구 변경을 반영할 수 있는 모델이다. 비용 승수는 초판 COCOMO 모델과 유사하나 몇 가지 새로운 요소가 도입되었다.

표 3.4 COCOMO II 모델의 세 가지 단계

비교 대상	단계 1 : 응용 합성 (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
유지 보수	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
프로덕트 비용 승수	없음	복잡도, 재사용 요구도	신뢰도, 데이터베이스 규모, 문서화 요구정도, 재사용 요구도, 제품 복잡도
플랫폼 비용 승수	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

프로토타입 구축에 필요한 노력을 추정하는 응용 합성 모델을 살펴보자. 응용 합성 모델은 컴포넌트 기반의 소프트웨어 개발 프로젝트에 많이 사용된다. 컴포넌트 기반 시스템은 그래픽 유저 인터페이스(GUI) 빌더, 데이터베이스 관리자, 미들웨어, 트랜잭션 처리, 미디어 관리자, 데이터 검색기 및 응용 분야 고유의 컴포넌트를 포함하고 있다. 응용 합성 모델을 사용하여 노력을 추정하는 과정은 다음과 같다.

1. 애플리케이션을 구성하는 화면, 보고서, 3세대 언어 컴포넌트의 숫자를 센다.
2. [표 3.5] (a)를 이용하여 화면과 보고서의 복잡도 수준을 결정한다.
3. [표 3.5] (b)에 있는 표를 이용하여 화면과 보고서, 3세대 언어 컴포넌트를 위한 복잡도 가중치를 찾는다.
4. 화면, 보고서, 3세대 언어 컴포넌트의 개수에 가중치를 곱하여 객체 점수(Object Point)를 계산한다.
5. 재사용률(reuse)을 예측하여 다음 공식에 대입하면 NOP(New Object Point)를 구할 수 있다.

$$NOP = OP \times (100 - \text{Reuse})/100$$

6. [표 3.4] (c)를 이용하여 객체 점수 생산성(PROD)을 결정한다.
7. 객체 점수 생산성을 식 PM = NOP/PROD 에 대입하여 최종 PM(Person Month) 값을 구한다.

표 3.5(a) 화면과 보고서의 복잡도 수준

화면				보고서			
포함된 뷰의 개수	자료 테이블의 개수			포함된 섹션의 개수	자료 테이블의 개수		
	〈 4	〈 8	8+		〈4	〈8	8+
〈 3	단순	단순	중간	0 또는 1	단순	단순	중간
3-7	단순	중간	복잡	2 또는 3	단순	중간	복잡
〉 8	중간	복잡	복잡	4 이상	중간	복잡	복잡

표 3.5(b) 복잡도 가중치

객체 타입	복잡도 가중치		
	단순	중간	복잡
화면	1	2	3
보고서	2	3	8
3세대 언어 컴포넌트			10

표 3.5(c) 객체 점수 생산성

개발자의 경험, 능력 개발 도구 경험, 성능	매우 낮음	낮음	중간	높음	매우 높음
PROD(NOP/Month)	4	7	13	23	30

사례 ❷ COCOMO II를 이용한 노력 예측

어떤 프로젝트의 계획 단계에 시스템의 일부를 프로토타이핑 하려고 한다. 프로토타입은 3개의 화면, 4개의 보고서가 필요하고 3세대 언어 컴포넌트는 사용되지 않는다. 각 화면은 뷰가 1개, 서버 자료 테이블이 1개 포함된다. 한 보고서는 2개의 섹션이 있으며 자료 테이블은 사용하지 않는다. 다른 보고서 2개는 각각 4개 이상의 섹션이 있으며 4개의 서버 테이블을 접근한다. 프로토타입의 50%는 이미 존재하는 컴포넌트를 재사용할 것이다. 개발 환경은 중간급이며 응용 도메인에 대한 경험은 거의 없다고 한다.

네 개의 화면은 모두 단순형이므로 객체 점수 1점씩 총 4점. 첫 번째 보고서는 단순형이므로 객체 점수 2점. 다른 2개의 보고서는 객체 점수 8점씩. 총 16점. 합하면 22점이다. 재사용률 50%를 적용하면 NOP는 11점. 낮은 개발 경험(7점)과 보통의 개발 도구 성능(13점)의 평균은 (7+13)/2 = 10. PROD = 10. 따라서 노력 = NOP/PROD = 1.1MM 이다.

3.3.3 기능 점수

기능 점수(function point)는 소프트웨어 시스템이 가지는 기능을 정량화한 것이다. 원시 코드가 아직 작성되지 않은 상태에서는 정확한 라인수의 예측은 불가능하다. 따라서 일반적인 소프트웨어가 갖는 기능(예를 들면, 입력, 출력, 질의, 파일, 인터페이스)의 개수로 소프트웨어의 규모와 복잡도를 나타내고 이를 시스템 개발에 필요한 기간과 소요 인력 계산의 기초로 삼는 방법이다.

- 입력 개수 – 사용자가 시스템에 제공하는 입력 자료의 개수
- 출력 개수 – 시스템이 사용자에게 제공하는 출력의 개수.
- 질의 개수 – 사용자가 시스템의 특정 기능을 요청하는 데 필요한 대화형 질의의 개수.
- 파일 개수 – 시스템이 유지 보관하는 정보의 그룹. 파일의 개수는 응용 분야에 따라 매우 다르다. 비즈니스 자료 처리 응용 분야의 소프트웨어는 파일의 개수가 많다.
- 인터페이스 개수 – 다른 외부 시스템과의 인터페이스, 예를 들면 다른 시스템에서 만든 파일을 읽거나 통신 라인으로 자료를 전달받는 경우 해당된다. 다른 시스템에서 만든 파일을 읽는 기능이 있다면 입력 항목과 인터페이스 항목 모두 해당된다.

주의할 점은 입력과 출력 개수는 입출력 화면이나 파일 단위의 그룹 항목 개수를 세어야 한다는 점이다. 즉, 입력 화면 안에 있는 개별 항목은 세지 않는다.

결론적으로 기능 점수는 시스템이 사용자에게 제공하는 기능을 기초로 응용 소프트웨어의 규모를 측정한 것이다. 이 방법은 경험 중심적이며 여러 가지 실험 결과에 의하면 비즈니스 응용 분야의 소프트웨어 개발비용 산정에 정확하다고 한다. 기능 점수를 이용하여 비용을 산정하려면 생산성 메트릭이 있어야 한다. 즉, 단위 시간당 프로그래머의 생산성을 기능 점수로 표현한 자료가 있어야 한다.

■ 기본 개념

기능 점수(Function Point)는 총 기능 점수(Gross Function Point)와 처리 복잡도 보정계수(Processing Complexity Adjustment)를 곱한 것이다. $FP = GFP \times PCA$.

기능 점수를 구하는 방법은 다음 6 단계로 구성된다.

1. 다섯 가지 기능 분야에 해당되는 개수를 파악하여 [표 3.6]의 작업표에 채운다.

2. 다섯 각 기능에 대한 복잡도(단순, 중간, 복잡)를 결정하여 표시한다.
3. 총 기능 점수(GFP)를 구한다. 즉 각 기능 분야의 개수와 복잡도 가중치를 곱하고 총합을 구한다.

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

표 3.6 기능 점수 구하는 표

	기능 분야	개수	복잡도			
			단순	보통	복잡	FP=개수×가중값
1	입력(트랜잭션)		3	4	6	
2	출력(화면 및 출력 양식)		4	5	7	
3	질의		3	4	6	
4	파일		7	10	15	
5	응용 인터페이스		5	7	10	
					GFP	

4. 다음 [표 3.7]에 있는 14개의 질문을 이용하여 각 처리 복잡도의 정도에 따라 0에서 5까지 할당한다. 0은 영향 없음, 1은 미미, 2는 약간, 3은 보통, 4는 상당, 5는 많음이다.

표 3.7 처리 복잡도 계산표

특 성	처리 복잡도
① 시스템이 신뢰도 높은 백업과 복구를 요구하는가?	
② 데이터 통신이 필요한가?	
③ 분산 처리 기능이 있는가?	
④ 성능이 중요한가?	
⑤ 시스템이 과부하 운용 환경에서 실행되는가?	
⑥ 온라인 데이터 입력이 필요한가?	
⑦ 온라인 입력이 다중 화면 위에 구축되기 위하여 입력 트랜잭션이 필요한가?	
⑧ 마스터 파일이 온라인으로 갱신되어야 하나?	
⑨ 입력, 출력, 파일, 질의가 복잡한가?	

⑩ 내부 처리가 복잡한가?	
⑪ 재사용 되도록 설계 하여야 하나?	
⑫ 변환과 설치가 설계에 포함되어 있나?	
⑬ 다중 사이트에 설치되기 위하여 설계되었나?	
⑭ 변경과 사용이 쉽도록 설계되었나?	

5. 처리 복잡도 보정계수(PCA)를 다음 식을 이용하여 구한다.

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

여기서 0.65는 경험에 의한 상수값이며 PCA 값은 0.65에서 1.35를 넘지 않는다.

6. 다음 식에 넣어 기능 점수를 구한다.

$$FP = GFP \times PCA$$

사례 ❸ 기능 점수 기법을 이용한 노력 추정

개발하여야 할 소프트웨어는 다음과 같은 기능 분야로 구성될 것으로 추정된다. 사용자 입력 = 10개, 사용자 출력 = 5개, 사용자 질의 = 8개, 자료 파일 = 30개, 외부 인터페이스 = 4개, 모두 복잡도는 단순이다.

처리 복잡도를 계산하기 위한 14개의 질문 중 신뢰도 높은 백업, 사용 친근성은 매우 높이 요구되며 나머지는 보통이다(PC1=5, PC14=5). 개발팀의 생산성은 주당 60 기능점수를 구현한다. 이 프로젝트를 수행하기 위한 노력을 추정하라.

먼저 [표 3.6]의 빈칸에 개수와 복잡도를 넣어 GFP를 계산하면

$$GFP = 10 \times 3 + 5 \times 4 + 8 \times 3 + 30 \times 7 + 4 \times 5 = 304 FP$$

처리 복잡도가 보정된 PCA를 계산하면

$$PCA = 12 \times 3 + 2 \times 3 = 42$$

여기에서 기능 점수 FP를 계산하면

$$FP = GFP \times PCA = 304 \times 42 = 12768 FP$$

마지막으로 추정된 노력은

$$E = FP / \ Productivity = 12768 / 6233 person - weeks$$

기능 점수는 구현되는 언어에 관계없는 메트릭이다. 흥미 있는 사실은 기능 점수 1점을 구현하기 위한 각 언어의 원시 코드 라인수는 크게 다르다는 것이다. 어셈블리 언어는 324, C 언어는 150, Pascal은 91, Ada는 71, APL은 32, Smalltalk은 21개의 원시 코드가 필요하다. 이런 수치는 프로그래밍 언어의 표현 능력을 상대적으로 나타내기도 하며 기능 점수를 기초로 최종 시스템의 원시 코드 라인수를 계산하는 데 사용할 수도 있다.

기능 점수 방법은 모든 항목에 일률적인 가중치가 적용되므로 문제가 있을 수 있다. 예를 들어, 단순한 처리로 가능한 입력과 복잡한 처리가 필요한 입력이 같이 취급된다는 문제점이 있다. 이런 문제점은 각 항목의 가중치를 하나의 값으로 하지 않고 범위를 정하여 단순, 평균, 복잡 등 가변적으로 만들어 적용함으로써 해결할 수 있다. 앞서 설명한 COCOMO II의 초기 설계 모델은 복잡도를 결정하는 기준을 정확히 제시하고 있다.

■ 국내 기능점수 산정 가이드

한국소프트웨어 산업협회에서는 국내 소프트웨어 개발 사업에 맞는 대가 산정 기준을 매년 제시하고 있다. 대가 산정 기준의 큰 틀은 COCOMO II의 초기 설계 단계의 모델을 따르고 있다. 각 기능의 타입은 다음과 같이 구분한다.

- 외부 입력(External Input) – 내부 파일의 내용에 영향을 주는 사용자 데이터 또는 제어 입력을 트랜젝션 단위(입력, 수정, 삭제)로 카운트한다.
- 외부 출력(External Output) – 소프트웨어 외부로 표출되는 사용자 데이터 또는 제어 출력을 트랜젝션 단위로 카운트한다(예를 들면 계산이나 통계 그래프가 있는 보고서).
- 내부 논리 파일(Internal Logical File) – 소프트웨어에 의하여 생성, 사용, 관리되는 파일을 포함하여 시스템에 존재하는 사용자 데이터 및 제어 정보의 그룹을 카운트한다.
- 외부 인터페이스 파일(External Interface File) – 소프트웨어 시스템 사이에 전달되거나 공유된 파일을 카운트한다.
- 외부 조회(External Query) – 중간 출력을 생성하는 외부 조회를 카운트한다.

각 기능 요소를 카운트 한 후에는 [표 3.8]을 이용하여 복잡도 수준과 가중치를 계산한다.

표 3.8(a) 기능 유형별 복잡도 판별표

내부논리파일과 외부 인터페이스				외부 출력과 조회				외부 입력			
레코드의 개수	자료 요소의 개수			파일 타입 개수	자료 요소의 개수			파일 타입 개수	자료 요소의 개수		
	1–19	20–50	50+		1–5	6–19	20+		1–19	20–50	50+
3	단순	단순	중간	0–1	단순	단순	중간	0–1	단순	단순	중간
2–5	단순	중간	복잡	2–3	단순	중간	복잡	2–3	단순	중간	복잡
6+	중간	복잡	복잡	4+	중간	복잡	복잡	3+	중간	복잡	복잡

표 3.8(b) 기능 유형별 복잡도 가중치

기능 타입	복잡도 가중치		
	단순	중간	복잡
내부 논리적	7	10	13
외부 인터페이스	5	7	10
외부 입력	3	4	6
외부 출력	4	5	7
외부 조회	3	4	6

사례 ❹ 국내 기능 점수 산정 가이드를 이용한 노력 예측

어떤 시스템의 사용자 지원을 위한 '서비스 가이드' 서브시스템을 개발하려 한다. 서브시스템의 기능은 도움말, 사용자 질문 등록, 사용자 질문 응답, FAQ, 자료실이다. 기능 점수 방법을 이용하여 노력을 추정해 보자.

개발하여야 할 소프트웨어는 다음과 같은 기능 분야로 구성될 것으로 추정된다. 내부 논리 파일 = 4개(FAQ는 필요 없음), 외부 인터페이스 파일 = 없음, 외부 입력(입력, 수정, 삭제 각 5개) = 15개, 외부 출력 = 없음, 외부 조회 = 5개.

먼저 각 기능요소에 대하여 표 3.8(a)를 이용하여 복잡도를 판별하여야 한다. 복잡도를 판별하기 어려운 경우 국내 기능 점수 산정 가이드에서는 기능 유형별 평균 복잡도를 다음과 같이 적용하여 간이법으로 계산할 수 있다.

표 3.9 복잡도를 고려한 기능점수(간이 계산)

기능 타입	UFP		
	개수	복잡도 평균	기능 점수
내부 논리적	4	7.5	30
외부 인터페이스	0	5.4	0
외부 입력	15	4	60
외부 출력	0	5.2	0
외부 조회	5	3.9	19.5
		UFP	109.5

구해진 기능점수에 국내 평균 생산성, 즉 기능 점수 당 투입공수를 곱하면 총 노력 산정값이 나온다. 2019년 한국소프트웨어 산업협회의 대가산정 가이드에 의하면 국내 평균 생산성은 표 3.10과 같다.

표 3.10 국내 기능 점수 당 평균생산성

구 분	예외값 하한	평균생산성	예외값 상한
생산성(FP/MM)	21.8	23.4	25.6

이 프로젝트를 위한 노력은 평균 노력을 구하면

$$E = FP/\ Productivity = 109.6/23.4 = 4.65MM$$

3.4 프로젝트 팀 조직

소프트웨어는 다른 제품과 달라서 프로젝트 팀을 구성하는 방법이 프로젝트 운영과 소프트웨어 결과물에 중요한 영향을 미친다. 예를 들어, 개발팀을 직능별로 나누어 분석 팀, 설계 팀, 개발 팀, 운영 팀, QA 팀, DB 팀 등으로 부서를 나눌 수 있다. 또는 각 직능을 담당하는 역할이 모여 프로젝트 별로 부서를 형성할 수도 있다. 조직 구성의 단위가 되는 부서를 어떻게 조직하고 부서 사이의 관계를 어떻게 정의하는가는 프로젝트 의사 결정과 업무 담당자들의 협력에 크게 영향을 주어 결국 소프트웨어 결과물이 달라질 수 있다.

프로젝트 팀 조직은 다음 세 가지를 정하는 일이다.

- 역할과 책임이 어디에 있는가?
- 어떤 통로로 정보가 전달되고 결정되는가?
- 어떻게 갈등을 해소할 것인가?

소프트웨어 개발을 위한 인력 자원과 작업 목표가 주어졌을 때 각자의 임무와 역할이 무엇인지 결정하여야 한다. 단체 운동 경기에서 잘 조직된 팀은 각자의 역할이 잘 분담되어 협력한다. 공을 한 선수가 잡았을 때 다른 선수는 다음에 패스를 받기 위하여 자기가 어떻게 할 것인지 역할을 잘 알고 뛴다. 즉, 잘 조직된 팀은 협력의 패턴을 보이는 것이 뚜렷하다.

3.4.1 팀 역할

소프트웨어를 개발하는 팀에 필요한 역할은 기술의 발전과 함께 매우 다양하고 빠르게 변해간다. 최근까지 소프트웨어 개발 조직에서 일반적으로 분류하는 직능은 다음과 같다.

- 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

3.4.2 직능별 조직

개발자들의 역할에 따라 같은 부서에 속하게 하는 것이 직능별 팀 조직이다. 직능별 팀 조직은 전통적인 방법으로 인력을 효율적으로 운영할 수 있고 전문 지식을 공유할 수 있어 개인의 기술이 잘 발전되고 기업 노하우로 축적될 수 있다.

직능별 팀 조직은 서로 다른 부서가 한 프로젝트의 다른 단계에 들어와 작업을 수행하고 나간다. 예를 들어 분석 팀이 요구를 추출하여 명세서를 완성하면 다음에는 아키텍트 팀이 들어와 설계한 후 개발 팀이 구현하게 된다. 즉 부분적으로 완성된 결과물이 한 부서에서 다른 부서로 전달되면서 소프트웨어가 진화해 나가는 것이다.

팀원은 한 부서에 소속되어 있고 지시받고 보고하여야 할 관리자가 한 사람이므로 안정적이고 잘 보호될 수 있다. 하지만 프로젝트의 협력은 부서별로 이루어져 한 부서의 작업 결과가 여러 다른 부서에 전달되어야 하므로 문서나 회의 등 의사 전달 체계가 잘 확립되어야 한다.

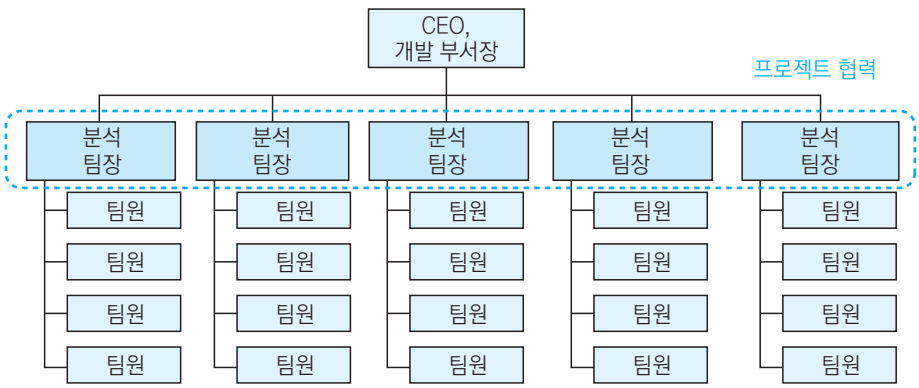


그림 3.11 직능별 프로젝트 조직

3.4.3 프로젝트별 조직

서로 다른 역할을 담당하는 직능별 개발자들이 프로젝트에 배정되어 프로젝트 별로 부서를 조직하는 방법이다. 즉 개발자가 어떤 프로젝트에 투입되면 시작부터 마칠 때까지 그 프로젝트 부서에서 일하는 것이다. 따라서 같은 팀이 소프트웨어 개발 주기의 모든 작업을 담당한다.

직능별 조직은 부서 사이에 생명 주기 각 단계의 커뮤니케이션이 필요한 반면 프로젝트별 조직은 프로젝트 팀 내부에서 작업 결과의 전달과 협력이 이루어진다. 따라서 의사 전달 경로가 짧으며 인력, 진도 등 프로젝트 관리가 수월하다.

프로젝트별 조직은 다양한 인력 자원이 프로젝트 안에 배치되어 있어 프로젝트 관리자가 독립성을 가지고 관리하며 조정할 수 있다. 따라서 고객 입장에서도 개발팀과 접촉

하기가 쉽다.

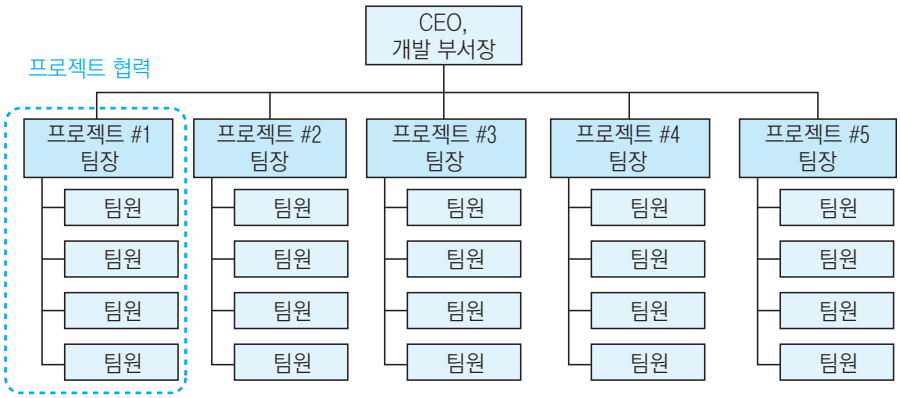


그림 3.12 프로젝트 별 조직

표 3.11 직능별 조직과 프로젝트별 조직의 비교

비교 대상	직능별 조직	프로젝트별 조직
장점	효과적인 인력 사용 전문성 유지 및 개발 용이 기술 전수 용이 공동 협력적, 안정적, 보안적 개인 소통 덜 요구됨	일정 및 비용 관리 용이 고객과의 연결이 간단 빠른 회신 커뮤니케이션 라인이 간단 관리, 교육이 용이 프로젝트 파악이 쉬움
단점	고객 인터페이스 약함 프로젝트 관리 취약 수평적인 소통이 취약 회신이 느림	기술적 불확실성 전문가 사용이 비효율적 미래 작업의 배정이 불투명 프로젝트 사이의 기술적 정보 교류 취약

3.4.5 매트릭스 조직

매트릭스 조직은 프로젝트별 조직과 직능별 조직의 구성을 혼합한 것이다. 즉 관리자와 개발자는 직능별 조직에 속해 있되 전문지식이 필요한 지정된 프로젝트를 위하여 간헐적으로 일한다. 직능별 조직의 관리자가 프로젝트 책임을 맡고 직능별 조직 부서에 소속된 개발자가 프로젝트에 참여하는 것이다.

매트릭스 조직의 중요한 특징은 모든 팀원이 보고하는 두 명 이상의 관리자(직능별 팀 관리자 및 프로젝트 관리자)가 있다는 것이다. 다수의 프로젝트를 수행하는 경우 보고

할 관리자가 더 많을 수도 있어 관계가 복잡해지고 커뮤니케이션 스킬이 더 많이 요구된다.

매트릭스 조직은 어떻게 운영하느냐에 따라 두 가지로 나눌 수 있다. 강한 형태의 매트릭스 조직은 프로젝트 관리자가 프로젝트에 책임을 지고 팀원들은 직능별 조직보다 프로젝트에 더 개입하는 형태이다. 반면에 약한 형태의 매트릭스 조직은 프로젝트의 책임을 직능별 관리자에게 맡기고 직능별 조직에 소속되어 있는 팀원도 여러 프로젝트에 적극적으로 담당하는 형태이다.

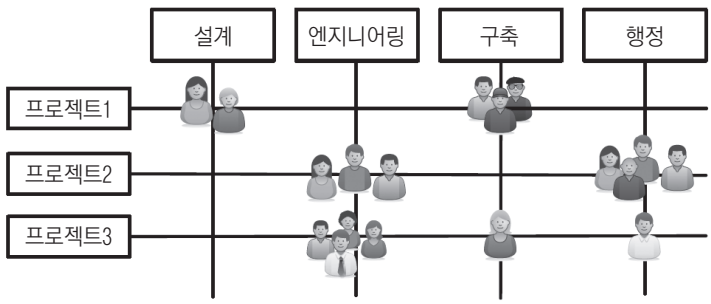


그림 3.13 매트릭스 조직

프로젝트 조직이 프로젝트 관리에 주는 영향

프로젝트는 조직 구조 유형에 의해 영향을 받는다. 그 영향을 두 가지 측면, 프로젝트 관리자 권한과 자원 사용 측면에서 따져 볼 수 있다.

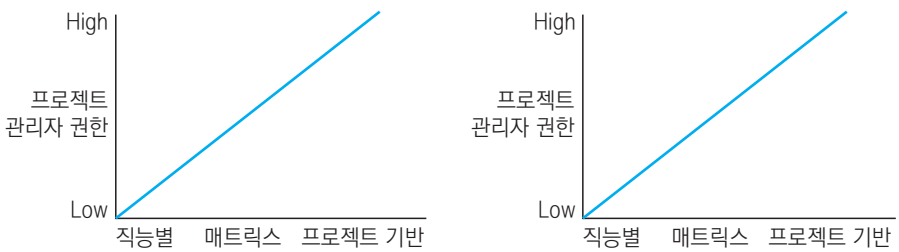


그림 3.14 조직 형태에 대한 두 가지 측면의 영향

직능별 조직에서 매트릭스, 프로젝트별 조직 구조 유형으로 갈수록 관리자의 권한은 계속해 왔던 직능적인 업무에서 배정된 프로젝트로 이동한다. 직능별 조직에서 관리자는 그 부서에 한정된 권한만 갖는다. 따라서 프로젝트를 성공적으로 수행하려면 전문가나 다른 직능 조직에 의지하고 협력하는 협상력이 있어야 한다.

자원 가용 측면에서는 직능별 조직, 매트릭스, 프로젝트별 조직으로 갈수록 더 큰 범위의 자원 동원이 가능하다. 직능별 조직은 인적 자원이 제한적이나 메트릭스나 프로젝트별 조직으로 가면 다양하고 효과적으로 제어할 수 있다.

3.4.6 애자일 조직

애자일 방법론은 선언문에 나와 있듯이 구성원의 상호작용과 고객과의 협력, 요구의 변경을 중요하게 여긴다. 따라서 애자일을 따르는 개발은 조직을 서로 밀접하게 협력하는 5~9명의 작은 인원의 팀으로 구성한다. 개인보다는 팀을 중요하게 생각하므로 책임과 역할을 서로 교환할 수도 있다.

애자일 조직에서 중요한 개념은 결과와 이슈에 대한 오너쉽을 공유한다는 것이다. 전통적인 조직에서 중요하게 생각하는 자원 관리과 리더십보다는 고객의 요구와 가치를 만족시키기 위하여 역동적이고 융통성 있고 최적화 된 조직을 구성한다. 팀원이 여러 직능을 담당할 수도 있고 필요에 따라 팀을 재구성할 수도 있으며 스스로 책임을 지고 관리하는 독립적 부서라 할 수 있다.

대표적인 애자일 방법인 스크럼에서는 다음과 같은 역할을 정의하고 있다.

- 스크럼 마스터 – 프로젝트 진도를 측정하고 문제를 해결한다. 또한 외부의 영향으로부터 팀을 보호하는 역할을 한다.
- 고객 – 스크럼에서는 고객도 개발 조직에 포함되어 요구분석 결과물의 오너가 된다.
- 팀 – 작업을 수행하기 위하여 스스로 조직하고 책임을 진다.

애자일 팀의 단점은 너무 작은 인원으로 구성되어 규모가 큰 프로젝트에는 맞지 않는다는 것이다. 이런 단점을 극복하기 위하여 애자일 팀을 여러 개 구성하는 경우도 있다.

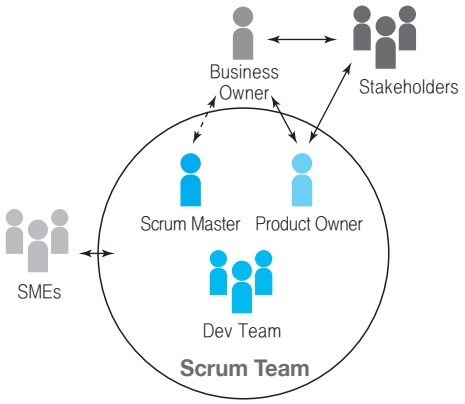


그림 3.15 애자일 조직

3.5 실행과 모니터링

만일 계획이 실행되지 않거나 실행에 도움이 되지 않는다면 의미가 없다. 프로젝트를 실행하는 동안 모니터링하고 현재 상황을 반영하기 위하여 계획과 비교하고 때로는 변경되어야 한다.

3.5.1 프로젝트 실행

프로젝트를 실행하면 다음 두 가지 관리 작업이 시작된다.

1) 작업 시작 미팅 – 작업이 시작됨을 공식적으로 선포하고 팀이 작업의 목표와 형식에 모두 잘 맞추어져 있는지 확인한다. 이를 위하여 팀원들과 소통하기 위하여 회의를 진행한다.
2) 작업 결과 수집 – 프로젝트의 결과물을 체계적으로 수집하기 위한 목적의 작업이다. 또한 배운 교훈을 평가하기 위한 기회가 된다. 프로젝트의 결과물을 수집하는 도구는 보관함(repository)과 버전화이다. 배운 교훈을 나누는 미팅이 작업 결과를 수집하는 공식적인 자리가 될 수 있다. 진행 상황에 대한 정보는 정기적으로 수집할 수도 있고 필요할 때 수집하는 경우도 있다. 모니터링하기 위하여 정량적인 데이터를 수집하고 작업 상황이나 난이도 등에 대한 느낌, 즉 정성적인 정보도 수집할 수도 있다.

3.5.2 프로젝트 모니터링

모니터링 하는 목적은 프로젝트의 현황을 파악하고 차이를 분석하여 필요하다면 초치를 취하려는 것이다. 현황은 범위, 일정, 비용, 품질 등의 관점으로 파악한다. 또한 데이터를 수집하여 분석하면 미래의 프로젝트에 대하여 더욱 정확한 계획이 가능하다.

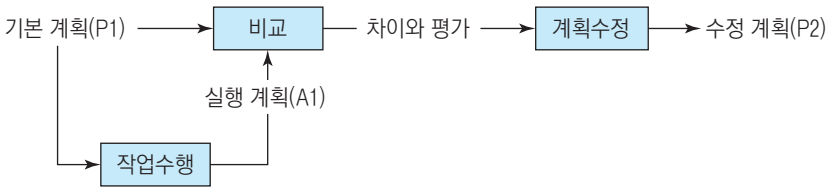
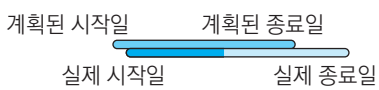


그림 3.16 모니터링

모니터링을 중점적으로 하여야 할 부분은 일정과 비용, 진척도이다. 일정은 프로젝트가 예정보다 빨리 진행되고 있거나 지연되고 있는지 이해하여야 한다. 비용은 예산보다 적게 쓰는지 오버되고 있는지 모니터링한다. 일정, 비용을 각각을 모니터링하는 방법은 간단하지만 부족한 부분이 있다.

■ 일정 모니터링

일정을 모니터링하는 기본적인 개념은 계획된 일정, 즉 주어진 시각의 개획의 스냅샷을 기초로 실행 값을 비교한다. 일정의 실행 현황, 즉 실제 시작한 날, 종료한 날, 실제로 들어간 노력과 진행상황을 비교한다.



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

그림 3.17 일정 모니터링

프로젝트에 투입된 노력을 시간단위로 환산하여 데이터를 수집할 수도 있다. 작업단위로 투입된 각 개발자의 노동 시간을 수집하여 합하면 주 또는 달별로 투입된 노력을 구

할 수 있다. 비용도 항목별 계획된 비용과 실제로 투입된 비용을 비교할 수 있도록 표로 만들어 모니터링 한다. 이런 방법은 직관적이고 간단하지면 프로젝트의 진행 상황을 한목에 볼 수 없는 단점이 있다.

■ 어닝 밸류 분석

비용과 일정을 통합된 방법으로 모니터링할 수 있는데 이를 어닝 밸류 분석(Earning Value Analysis)이라 한다. 어닝 밸류 분석은 과정이 복잡하지만 전제척인 진척 상황을 이해하기 쉬운 장점이 있다.

어닝 밸류 기법은 계획된 노력(비용), 실제 진적도(어닝 밸류), 노력(실제 비용)을 금전적 가치로 측정하여 통합된 모니터링을 제공하다. 즉 계획, 작업, 진척도를 같은 단위로 측정하여 비교가 가능하게 한다. 진척도를 노력으로 비교할 수 있고 여기에 예산과 비용이 녹아 있어 사용이 편리하다.

어닝 밸류의 기본적인 원리는 맨파워를 비용으로 본다. 계획된 비용과 실제 사용된 비용을 누적하여 표현하되 결과물의 달성도를 금액으로 환산한 가치로 나타낸다. 어닝 밸류는 그림 3.19와 같은 그래프로 나타낼 수 있고 계획 단계와 실행 단계를 비교하여 진행 상황의 추이를 확인할 수 있고 미래의 비용과 일정을 예측할 수 있다.

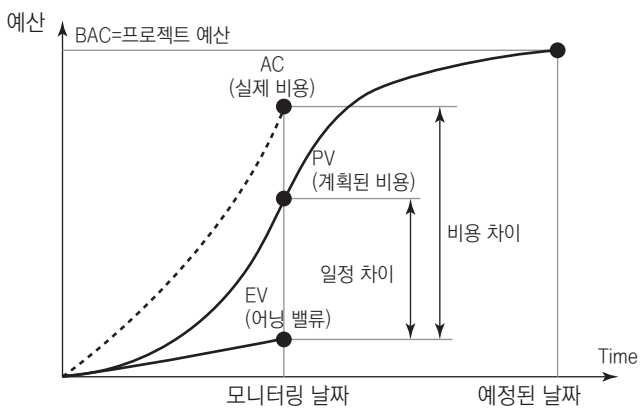


그림 3.18 어닝 밸류 분석

3.5.3 번다운 차트

애자일 프로세스 중 스크럼에서는 수행된 작업보다는 남아 있는 작업에 초점을 두어

일정을 모니터링 한다. 스크럼에서는 구현될 각 기능에 이상적인 투입 시간을 할당한다. 각 기능은 스프린트에 배정되어 기능이 완성되면서 소멸되는 스프린트 점수로 기록한다.

번다운 차트의 목표는 기능이 출시되는 속도를 측정하는 것이다. 이상적인 번다운과 실제 번다운의 차이를 보여주는 방법인데 스프린트 사이의 속도는 일정하다는 전제가 있다.

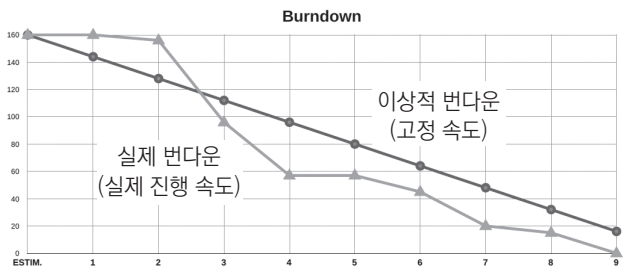


그림 3.19 번다운 차트

3.6 리스크 관리

프로젝트 계획서를 보면 프로젝트는 이러 이러한 목적을 달성하여야 한다고 쓰여 있다. 하지만 계획 단계에는 목적 달성을 위한 다양한 불확정성을 다루어야 한다. 계획은 명시되어 있으나 성취하기 위한 현실은 변화와 부정적 요인이 많기 때문이다.

리스크 관리란 위험 요인을 파악하고 평가 관리하는 기술과 노하우, 프로세스를 의미한다. 리스크 관리의 목적은 프로젝트에 대한 긍정적인 사건이 일어날 수 있는 가능성을 높이고 부정적인 사건이 일어날 수 있는 가능성을 낮추기 위한 것이다.

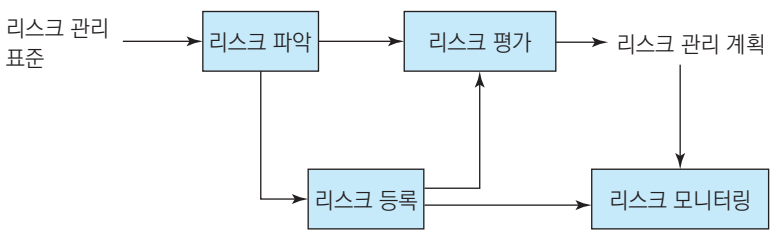


그림 3.20 리스크 관리

3.6.1 리스크 파악

프로젝트에 기본적으로 영향을 줄 수 있는 리스크가 무엇인지 이해하고 특징을 문서화한다. 리스크 파악은 반복적인 작업이며 리스크를 등록하여 정량, 정성적인 평가의 기초 자료가 된다. 리스크를 찾는 방법은 다음과 같다.

- 회의 – 리스크 파악을 위한 회의를 통하여 아이템을 모은다.
- 문서 분석 – 프로젝트와 관련된 문서를 읽고 리스크를 찾아낸다.
- 리스크 분할 구조, 체크리스트 – 리스크 아이템을 분할하여 계층 구조로 그리거나 체크리스트를 만들어 파악한다.
- 유추 – 비슷한 프로젝트 유형과 비교하여 유추한다.

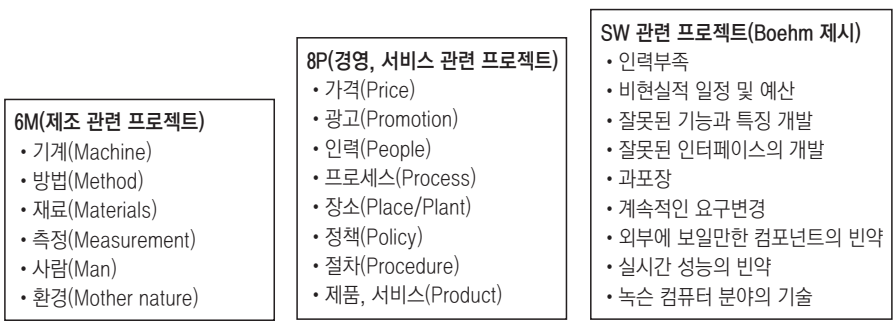


그림 3.21 리스크 요소

3.6.2 리스크 평가

리스크를 프로젝트에 대한 영향도에 따라 평가하고 우선순위를 매긴다. 리스크는 얼마나 자주 발생하는가에 대한 확률과 리스크가 발생했을 때 미치는 영향에 따라 우순순위를 매길 수 있다.

평가하는 방법은 주관적인 판단으로 평가하는 정성적인 방법과 리스크 확률을 고려한 영향을 돈으로 환산하는 정량적인 방법이 있다.

정성적인 방법은 리스크에 대한 확률에 대한 정확한 정보가 없을 경우 사용한다. 정성적인 분석 방법은 확률과 영향도에 대하여 채택된 스케일로부터 출발하여 그림 3.23과 같은 리스크 매트릭스에 리스크 요소를 표시하면 된다.

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

리스크 매트릭스에서 발생 빈도가 높고 영향이 큰 것(E)은 특별히 신경을 써서 관리하여야 하며 그렇지 못할 경우 실패로 끝날 가능성이 많은 요소다. 그 다음으로 중요한 리스크(H)는 면밀히 모니터링 하여야 할 리스크이다. 그 이외의 요소들은 일반적으로 해가 되는 요소들이다.

3.6.3 리스크 관리

리스크를 관리한다는 것은 받아들일 수 없는 위험 요소를 해소하기 위한 방법을 강구하고 프로젝트 실행하는 동안 이를 적용하는 작업이다.

E와 같이 위협적인 요소에 대한 관리 전략은 리스크를 피하기 위하여 계획을 바꾸거나 책임을 다른 기관에 맡기는 방법이 있다. 또는 빈도나 영향도를 낮추기 위하여 프로토타이핑 할 수도 있다. 리스크를 기회로 바꾸는 계기가 될 수도 있는데 불확실성을 제거하기 위하여 유능한 인재를 등용하거나 이익을 나누기 위하여 제3자와 협업할 수도 있다.

리스크 요소는 계속 관찰하여 계획과 차이가 난다면 그 원인을 찾아내어 해결 방안을 제시하거나 계획을 수정한다. 미리 인지된 리스크는 긴급대책 계획이 있어 대처하기가 쉽다. 하지만 예상하지 못한 리스크는 인식하고 분석하는 모든 과정의 작업이 필요하다.

리스크 관리에서 유념할 점은 완전히 큰 위험 요소는 파악할 필요가 없다. 그런 경우는 프로젝트 시작을 포기하면 된다. 또한 영향을 원인과 혼돈하지 않아야 한다. 프로젝트가 지연된다거나 지체보상금이 부과되는 것은 프로젝트에 미치는 영향이다.

프로젝트 계획서

프로젝트 계획서의 내용은 프로젝트의 성격이나 사용자의 요구에 따라 다양해질 수 있으나 그림 3.24의 차례가 계획서 안에 포함될 내용을 보여준다. 프로젝트 계획서에는 개요, 자원 및 일정 예측, 조직 구성, 작업 분해, 기술 관리 방법 등이 포함되어야 한다.

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. 기술 관리 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
8. 개발 환경
9. 성능 시험 방법
10. 문서화
11. 유지 보수
12. 설치, 인수
13. 참고 문헌 및 부록

그림 3.23 소프트웨어 개발 계획서의 목차

개요에는 프로젝트의 목적과 산출물 및 배경 등을 간략하게 기술한다. 또한 계획서에서 참조하는 모든 문서를 나열하고 미리 용어와 약어에 대하여 설명한다. 개요 부분은 계획서 상의 자세한 내용을 한 눈에 보아 알

수 있는 요약이 된다.

다음 장에는 소작업이 정의되며 각 작업 사이의 의존 관계를 나타낸다. 또한 프로젝트를 수행하기 위하여 어떤 자원이 필요하고 프로젝트의 소작업과 기능에 어떻게 자원을 배정할 것인가를 기술한다. 일정 계획도 포함시킨다. 또한 개발 조직의 구성도와 각 조직의 의무 및 기능이 기술되어야 한다.

또한 어떻게 제품을 개발할 것인가를 제품의 생명 주기라는 관점과 개발 조직의 구성이라는 두 가지 관점에서 기술한다. 프로세스 모델, 즉 소프트웨어의 개발을 위한 여러 가지 소작업(예를 들면, 설계, 통합 시험 등)과 프로젝트 기능(예를 들면, 프로젝트 관리, 형상 관리, 품질 관리 등)을 기술한다.

관리 계획으로 관리를 위한 철학, 목표, 우선순위들을 나열하며 이들은 필요성과 일정, 예산 형편, 위험도에 따라 분류한다. 프로젝트를 위하여 요구되는 인원수와 자질, 기간도 제시한다. 기술 계획으로 개발에 사용될 하드웨어나 소프트웨어, 운영 체제, 개발 방법, 팀 구성, 프로그래밍 언어, 도구, 기술이 포함된다. 또한 문서화나 코드 표준이 포함된다.

부록에는 외주 용역 관리 계획, 보안 계획, 검증 계획, 교육 계획, 하드웨어 교정 계획, 설치 계획, 제품 유지보수 계획 등을 첨부할 수 있다.

연습문제

1. 다음 중 계획에 대한 설명으로 바르지 못한 것은?

① 계획은 제한된 자원과 제한된 일정으로 결과를 생성하기 위한 방법을 모색하는 것이다.
② 계획은 노동 집약적인 개발을 지원하기 위하여 새로운 인력 자원을 찾는 것이다.
③ 계획은 보이지 않는 것을 찾고 조정하는 것이다.
④ 계획은 많은 사람의 노력을 융합하여 제품을 만들고 이를 통하여 고객의 요구를 만족시키는 것이다.

2. 다음 중 소프트웨어 계획단계에서 이루어지는 일이 아닌 것은?

① 소프트웨어 개발의 범위에 대한 정의
② 소프트웨어 개발을 위해 필요한 자원들의 예측
③ 소프트웨어 모듈 및 자료구조의 정의
④ 소프트웨어 개발을 위한 비용과 일정의 추정

2. 다음 중 계획 단계에 사용되는 기법에 대한 설명이 옳지 않은 것은?

① COCOMO 모델은 규모를 기반으로 노력을 추정하는 도구이다.
② 기능 점수는 소작업을 기반으로 노력을 추정하는 Bottom-up 방법이다.
③ WBS는 일정 계획 작업의 입력이 된다.
④ CPM 네트워크는 작업의 선후 병렬 수행 관계를 그린 것으로 최소 소요 기간을 파악할 수 있다.

4. 소프트웨어 프로젝트 계획에서 스케줄링을 위한 작업 순서를 바르게 나열한 것은?

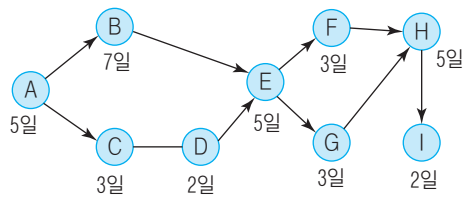
① 작업 소요시간 및 우선순위 결정 → 작업 분해 → CPM 네트워크 작성 → 임계 경로 추출 → 일정표 작성
② 작업 분해 → 작업 소요시간 및 우선순위 결정 → CPM 네트워크 작성 → 임계 경로 추출 → 일정표 작성
③ 작업 분해 → 작업 소요시간 및 우선순위 결정 → 일정표 작성 → CPM 네트워크 작성 → 임계 경로 추출
④ 작업 소요시간 및 우선순위 결정 → 작업 분해 → 일정표 작성 → CPM 네트워크 작성 → 임계 경로 추출

5. 다음은 어떤 소프트웨어 프로젝트를 구성하는 작업들의 선행 작업과 소요기간을 나타낸 표이다. 이 프로젝트를 위한 액티비티 네트워크의 임계경로는? [7급 국가직 임용고시, 2015]

작업	T1	T2	T3	T4	T5	T6	T7	종료
소요기간(일)	6	4	2	3	5	3	2	
선행작업	-	-	T1	T2	T2, T3, T4	T3, T5	T5, T6	T7

① T1 → T3 → T6 → T7
② T1 → T3 → T5 → T6 → T7
③ T2 → T5 → T6 → T7
④ T2 → T4 → T5 → T6 → T7

6. 다음 CPM 네트워크에 대한 설명으로 옳지 않은 것은? [7급 국가직 임용고시, 2010]



① 프로젝트가 종료되는 데 소요되는 최소 시간은 32일이다.
② 프로젝트 일정 중 임계 경로는 A-B-E-G-H-I이다.
③ 여유 시간은 D가 제일 많으며 2일이다.
④ E작업의 최대로 빠르게 시작할 수 있는 시간은 12일 이후이다.

7. 다음은 소프트웨어 프로젝트의 소요 기간과 비용 추정에 대한 설명이다. 올바른 것은?

① 요구되는 개발 일정이 촉박할수록 COCOMO 모형의 노력 승수는 작아진다.
② 기능 점수 추정 방법은 LOC를 추정하기 위한 방법이다.
③ 기본형 COCOMO-81에서 같은 량의 노력이 추정되었을 때 가장 오래 걸릴 것으로 예상되는 소프트웨어의 타입은 중간형(semidetached)이다.
④ Putnam 추정 모델은 개발 노력과 개발 기간과의 관계는 선형적이라는 사실을 보여준다.

8. 다음은 COCOMO 모델의 노력 보정 계수(effort adjustment factor) 중 개발 요원의 특성에 대한 것이다. 만일 어떤 고용주가 소프트웨어 프로젝트를 수행하기 위해 앞의 정보만 이용하여 Java 프로그래머를 고용하고자 한다면 어느 것이 가장 옳은 판단일까?

비용 승수 요소	승수값					
	매우낮음	낮음	정상	높음	매우높음	극히높음
프로그래머 능력	1.42	1.17	1.00	0.86	-	
프로그래밍 언어 경험	1.14	1.07	1.00	0.95	0.70	

① Java 프로그래밍 언어의 경험이 프로그래머의 자질과 능력보다 더 큰 영향을 미치므로 Java 언어의 능력이 우선되어야 한다.

② 프로그래머의 자질과 능력이 특정 언어의 경험보다 더 큰 영향을 미치므로 프로그래머의 자질과 능력을 우선 고려하여야 한다.

③ 프로그래머의 능력이 낮을수록 프로그램의 규모가 늘어날 수 있어 교육을 고려하여야 한다.

④ 프로그래밍 언어 경험이 높을수록 비용이 적게 들어 좋은 대우를 고려하여야 한다.

9. COCOMOII의 서브모델이 아닌 것은?

① 초기 설계 모델(early design model)

② 임베디드 모델(embedded model)

③ 응용 결합 모델(application composition model)

④ 재사용 모델(reuse model)

10. 기능점수(FP)를 계산하기 위해 고려할 대상으로 옳지 않은 것은?

① 외부조회(EQ)

② 내부논리파일(ILF)

③ 외부연계파일(EIF)

④ 내부출력(IO)

11. 기능 점수(Function Point)에서 EI 중의 하나인 회원 삭제 기능의 복잡도를 산정하기 위해 DET 개수를 산출하려고 한다. ILF인 회원 정보는 회원 번호, 회원 이름, 연락처, 주소, 이메일 필드로 구성된다. 회원 삭제는 회원 번호와 기능 키(Ctrl+R)를 선택하여 이루어진다. 만약 삭제하려는 회원 번호가 회원 정보에 존재하지 않는다면, 오류메시지가 출력된다. 산출되는 DET 개수는?

① 3개 ② 4개
③ 5개 ④ 6개

12. 어떤 소프트웨어의 총 개발 기간은 12개월이며, 5명의 개발 인력이 처음에 투입되었다. 5개월 시작 시점에 추가로 3명이 투입되어 개발이 완료되었다면, 이 소프트웨어의 개발에 투입된 총 인-월(Man-Month)은?

① 75 ② 81
③ 84 ④ 96

13. 개발할 소프트웨어의 예측된 총 라인수가 5만 줄이며 개발할 인력은 5명이다. 평균 생산성이 500줄/MM이라면 개발에 소요되는 적정 기간은? [7급 국가직 임용고시, 2012]

① 10개월 ② 15 개월
③ 20개월 ④ 25개월

14. 다음은 어떤 프로젝트 조직의 장점을 나열한 것인가?

> 효과적인 인력 사용
> 전문성 유지 및 개발 용이
> 기술 전수 용이
> 공동 협력적, 안정적, 보안적
> 개인 소통 덜 요구됨.

① 직능별 조직 ② 프로젝트 별 조직
③ 강한 매트릭스 조직 ④ 약한 매트릭스 조직

15. 프로젝트 팀 조직이 프로젝트에 주는 영향을 두 가지 측면(관리자 권한과 자원 사용 측면)으로 평가하였을 때 옳은 것은?

① 직능별 조직, 매트릭스, 프로젝트별 조직 순서로 권한과 자원 사용이 강하다.
② 직능별 조직, 매트릭스, 프로젝트별 조직 순서로 권한과 자원 사용이 약하다.
③ 직능별 조직, 프로젝트별 조직, 매트릭스 순서로 권한과 자원 사용이 강하다.
④ 직능별 조직, 프로젝트별 조직, 매트릭스 순서로 권한과 자원 사용이 약하다.

16. 위험(risk)에 관한 설명으로 옳지 않은 것은?

① 위험은 실제로 발생하여 프로젝트 결과에 부정적인 영향을 끼친 문제이다.
② 프로젝트 위험(project risk)은 프로젝트 일정이나 자원에 영향을 미치는 위험으로 경험 있는 개발인력이 도중에 그만두는 것이 그 예이다.
③ 제품 위험(product risk)은 개발될 소프트웨어의 품질 혹은 성능에 영향을 미치는 위험으로 구매한 컴포넌트가 예상대로 성능을 내지 못하는 것이 그 예이다.
④ 위험은 발생 가능성과 발생시 프로젝트에 미치는 영향의 정도로 계량화될 수 있다.

17. 프로젝트 리스크 관리에 대한 올바른 작업 순서는?

① 리스크 파악→리스크 등록→리스크 평가→리스크 모니터링 관리
② 리스크 파악→리스크 평가→리스크 등록→리스크 모니터링 관리
③ 리스크 파악→리스크 등록→리스크 모니터링 관리→리스크 평가
④ 리스크 파악→리스크 평가→리스크 모니터링 관리→리스크 등록

18. 프로젝트 실행 단계의 모니터링에 대한 설명 중 옳지 않은 것은?

① 모니터링 하는 목적은 프로젝트의 현황을 파악하고 차이를 분석하는 것이다.
② 프로젝트 일정은 투입된 노력을 시간단위로 환산하여 데이터를 수집하여 모니터링 할 수 있다.
③ 어닝 밸류 분석은 비용과 일정을 따로 모니터링하는 방법이다.
④ 번더운 차트는 스프린트에 배정되어 기능이 완성되면서 소멸되는 스프린트 점수로 표시한다.

19. 비용이 1억 5천만원에 6개월로 추정한 프로젝트가 있다. 3개월 후 어닝 밸류 분석은 다음과 같다.

EV = 6천5백만원
PV = 7천5백만원
AC = 8천만원

일정(SV)과 비용의 차이(CV)는 얼마인가?

① SV= +천만원 / CV= +천5백만원
② SV= +천5백만원 / CV= −천만원
③ SV= −5백만원 / CV= 천5백만원
④ SV= −$천만원 / CV= +천5백만원

20. 프로젝트 계획서 안에 포함되어야 할 내용으로 가장 거리가 먼 것은?

① 프로젝트의 목적과 산출물 및 배경
② 프로젝트를 수행하기 위한 일정과 자원
③ 변경, 위험 관리, 진도 관리 등 기술 관리 방법
④ 요구에 대한 제약 및 검증 방안

21. 프로세스를 시작할 것인지를 결정하는 두 가지 요인은 무엇인가?

22. 프로젝트 계획 수립에서 해야 할 작업은 무엇인가? 또한 프로젝트 계획을 수립할 때 고려해야 할 사항은 무엇인가?

23. 다음은 어떤 프로젝트에 대한 WBS를 나타낸 것이다.

작업	선행 작업	소요 기간(주)
A	없음	2
B	없음	6
C	없음	2
D	C	3
E	A	4
F	B, D	3
G	B, D	1
H	G	4
완성	E, F, H	

(1) CPM 네트워크를 그려라.
(2) 임계 경로를 찾고 그 경로를 네트워크 상에 그려라.
(3) 이 프로젝트를 위해 걸리는 최소 기간은?
(4) 이 프로젝트를 위한 간트 차트를 그려라.

24. 타당성 분석이란 무엇이며 포함되어야 할 내용에는 어떤 것들이 있는가?

25. COCOMO 모델이 무엇이며 COCOMO-81과 COCOMO II의 차이점을 설명하라.

26. 프로젝트 팀 조직의 세 가지 유형, 직능별 조직, 프로젝트별 조직, 매트릭스 조직은 무엇이며 각각의 장단점은 무엇인가?

27. 애자일 조직의 특징은 무엇이며 스크럼 조직의 역할은 무엇이 있는가?

28. 프로젝트 일정을 모니터링하기 위한 어닝 밸류 분석 방법이란 무엇인가? 또한 번다운 차트란 무엇이며 어떤 장점이 있나?

29. 다음 프로젝트에 대하여 적어도 5가지 이상의 위험 요소를 식별하고 그 해결 방안을 제시하라.

"콘텐츠 전문 회사에서 새로운 아이디어로 휴대폰의 사진기 앱을 만들려고 한다. 프로그래밍 전문 인력도 없으나 아이디어는 특허를 출원하여 프로젝트를 시작하였다. 사진기 앱은 용도에 따라 다른 버튼과 위젯을 다양화하려고 한다. 특히 버튼에는 캐릭터를 커스터마이징 할 수 있는 계획도 가지고 있다. 출시 시기는 아직 잡지 않았다."

3. 프로젝트 계획서 안에 포함되어야 할 내용은 무엇인가?