소프트웨어 공학개론

Introduction to Software Engineering

2022-1학기 (Spring)
선문대학교 AI소프트웨어학과

3월 22일 (화)

강의 목차

3. 프로젝트관리

- 3.1 프로젝트관리란?
- 3.2 개발공수의 산정
- 3.3 품질관리
- 연습 문제

3. 프로젝트관리

개요

- 대규모 및 고품질 소프트웨어의 정해진 예산과 기한내에 개발 위해
  - 소프트웨어 작성의 사전 계획 및 계획의 올바른 이행 필요
  - 프로젝트가 지연없이 진행될 수 있도록 해야 함
    - 사양, 개발기간, 개발예산, 개발인력 등이 도중에 변경되더라도
  - 프로젝트를 관리할 때
    - 소프트웨어개발프로세스의 각 공정에서 해당 공정의 전문가들이 고도의 지적활동을 수행하고 있음을 충분히 인식해 두는 것이 중요

- 이 장에서는 프로젝트관리의 개요에 대해서 설명
  - 프로젝트관리에서 개발공수 산정 방법
  - 소프트웨어개발 성과물 및 프로세스의 품질 평가 방법

3.1 프로젝트관리란?

개요

- 프로젝트관리
  - 요구사항을 만족하는 소프트웨어를 고객에게 정해진 기간내에 납품
    - 이를 위해서, 소프트웨어개발에서 누가 어떤 단계에서 무엇을 하면 좋은지를 계획하는 작업
  - 대책작업
    - 계획한대로 프로젝트가 진행되고 있는지 확인
    - 확인 결과에 따른 대책작업

3.1 프로젝트관리란?

선문대학교

목차

- 프로젝트관리의 개요
  - 3.1.1 프로젝트관리지식체계
    - 프로세스 분류
      - 1) 착수 → 2) 계획 → 3) 실행 → 4) 통제 → 5) 종료
  - 3.1.2 개발계획
    - 1) 개발목적 → 2) 개발목표 → 3) 개발대상업무 및 운용방침

      → 4) 개발시스템의 기본구성 → 5) 개발공수 → 6) 개발일정

      → 7) 개발체제 → 8) 개발환경 및 개발방법

      → 9) 성과물의 관리방법 → 10) 위험요소의 관리방법

3.1 프로젝트관리란?

선문대학교

3.1.1 프로젝트관리지식체계 (Project Management Body Of Knowledge : PMBOK)

- 기존 프로젝트관리에서의 3가지 관점: QCD(Quality: 품질, Cost: 비용, Delivery: 납기)
- 프로젝트관리지식체계
  - 프로젝트관리를 수행하는데 필요한 기본적인 지식의 체계화
  - 미국프로젝트관리협회 (Project Management Institute : PMI)
  - (그림) PMBOK의 9가지 관점

표 3..1 | PMBOK의 9가지 관점(지식영역)

관점	내용
통합관리	프로젝트계획의 책정, 실시, 변경관리
범위관리	범위(프로젝트에서 실시되는 작업 및 프로젝트에서 생성되는 성과물)에 관한 정의, 검증, 변경관리
시간관리	일정작성, 작업순서결정, 소요시간산정, 진척관리
비용관리	자원계획, 비용산정과 예산화
품질관리	품질(신뢰성, 사용성, 유지보수성 등)의 평가와 보증
인적자원관리	조직구성, 개발인력의 조달, 팀편성 및 육성
의사소통관리	정보의 공유화, 의사소통의 확립, 실적보고
위험요소관리	위험요소의 식별, 정량화, 대응책수립
조달관리	기기의 조달, 발주처의 선정, 계약

3.1 프로젝트관리란?

선문대학교

3.1.1 프로젝트관리지식체계

- 프로젝트의 프로세스 분류 (5가지)
  - (1) 착수프로세스
    - 프로젝트 또는 프로젝트 구성 단계 정의 및 인가
  - (2) 계획프로세스
    - 프로젝트의 목표와 범위 결정
    - 관련 활동 계획 및 수행 (예: 자원(예산, 인력, 기기 등)의 조달)
  - (3) 실행프로세스
    - 프로젝트계획 실행, 범위 검증, 품질 보증에 관한 활동 수행
  - (4) 통제프로세스
    - 프로젝트계획과의 차이점 인식, 실적보고 및 진척상황 측정 수행
    - 변경관리 및 위험요소에 대한 대응 활동 수행
  - (5) 종료프로세스
    - 계약종료절차 수행, 프로젝트 종료

3.1 프로젝트관리란?

3.1.1 프로젝트관리지식체계

- 일본
  - 소프트웨어개발에 관한 일련의 개발공정 및 거래 프로세스를 명확하게 할 목적
    - SLCP-JCF98
      - 벤더(제품을 제조 또는 판매하는 회사)와 고객 사이에서 공통적으로 참조할 수 있는 가이드라인
      - 이러한 가이드라인의 공통프레임 98(1998년) 규정항목
        - 6가지 주요 프로세스
        - 8가지 지원프로세스
        - 조직에 관한 4가지 프로세스
        - 시스템감사프로세스
        - 수정프로세스 등
- 한국
  - KS표준 규정
    - KSXISOIECTR16326
      - 소프트웨어 프로젝트관리에 적용하기 위한 지침

3.1.2 개발계획

- 프로젝트 관리를 위해
  - 단순히 진척상황을 추적하여 임기응변식으로 다음 작업을 결정하는 것으로는 불충분
  - 사전에 여러 가지 사항들을 고려하여 계획을 수립
  - 실제 상황과 계획 사이의 차이점을 파악할 필요가 있음
- 작성된 개발계획
  - 프로젝트의 멤버 및 고객 등이 개발대상 소프트웨어에 관한 정보와 의식을 공유하는데 있어서도 중요한 역할을 수행
- 소프트웨어개발을 수행할 때,
  - 필요한 자원을 적절한 시기에 투입 필요
- 개발전
  - 개발공수 및 개발비용을 산정 및 이에 맞추어서 자원 배치 준비 필요

3.1.2 개발계획

- (1) 개발목적
  - 무엇을 위하여 개발을 수행해야하는가, 무엇을 위하여 프로젝트를 추진해야 하는가 등과 같은 목적

- (2) 개발목표
  - 시스템이용자의 요청사항 및 업무운영상의 방침 등 개발할 소프트웨어를 도입함으로써 달성되는 목표

- (3) 개발대상업무 및 운용방침
  - 개발하려는 소프트웨어가 실제 업무에서 어떻게 사용되는지 등과 같은 개발대상의 범위와 기능
  - 업무상의 제약사항과 이용자에 관한 전제조건 등 포함
  - 운용 및 유지보수를 누가 어느정도 수행하는가 등에 관한 사항들 포함

3.1.2 개발계획

- (4) 개발시스템의 기본구성
  - 개발하려는 소프트웨어가 실제로 가동되는 환경(OS, 하드웨어, 네트워크 등) 및 가능한 한 아키텍처를 결정할 때의 전제조건(예: 클라이언트-서버 모델 등)

- (5) 개발공수
  - 개발에 필요한 공수와 개발비용, 고객과의 계약을 수행하기 위하여 또는 개발일정 및 개발체제를 확정하기 위하여, 개발공수와 개발비용을 산정할 필요가 있음
  - 과거에 수행했던 유사한 프로젝트의 사례 및 실적을 참고하여 수행하며, 산정작업을 완벽하게 수행하는 것은 불가능
  - 계획단계에서는 대략적인 산정을 하고, 개발 중간단계에서 반복하여 수정 필요

3.1.2 개발계획

- (6) 개발일정
  - 산정작업에 의해 얻어진 개발공수에 따른 각 개발단계에서 소비되는 기간
  - 개발일정을 기술하는 도구 : 작업명세구조, 간트차트
  - 작업명세구조(Work Breakdown Structure : WBS)
    - 프로젝트에서 실행되는 작업(Activity)들을 분할하고 상세화해서 기술한 것

그림 3.1 작업명세구조의 예

3.1.2 개발계획

- (6) 개발일정
  - 간트차트
    - 각 작업들을 병렬로 기술하고, 각각의 작업기간을 막대그래프형태로 나타낸 것
    - 현시점에서 어떤 작업이 실행중 또는 종료되고 있는지, 향후 어떤 작업이 수행될 예정인지 한눈에 알 수 있음

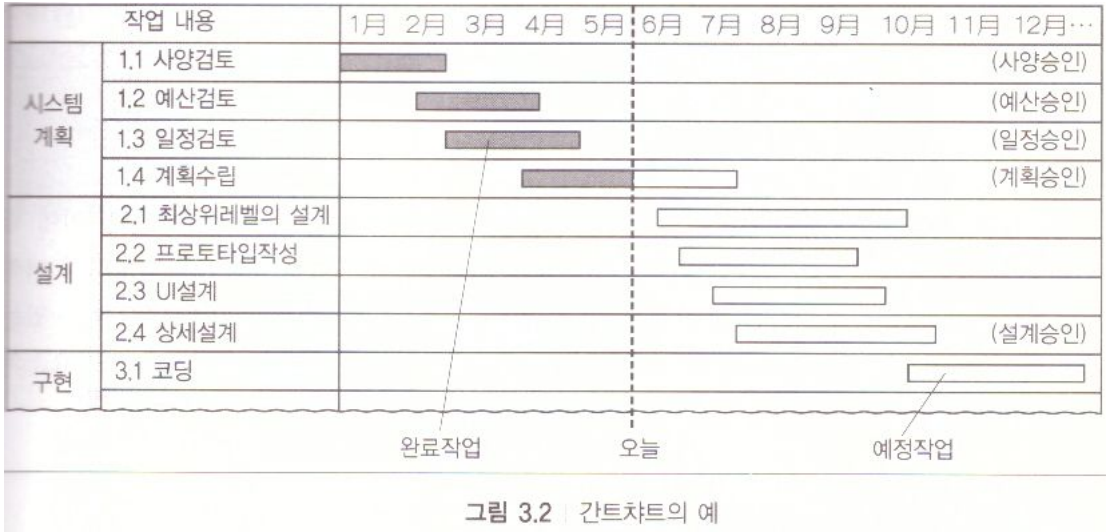


그림 3.2 간트챠트의 예

3.1.2 개발계획

- (6) 개발일정
  - 간트차트

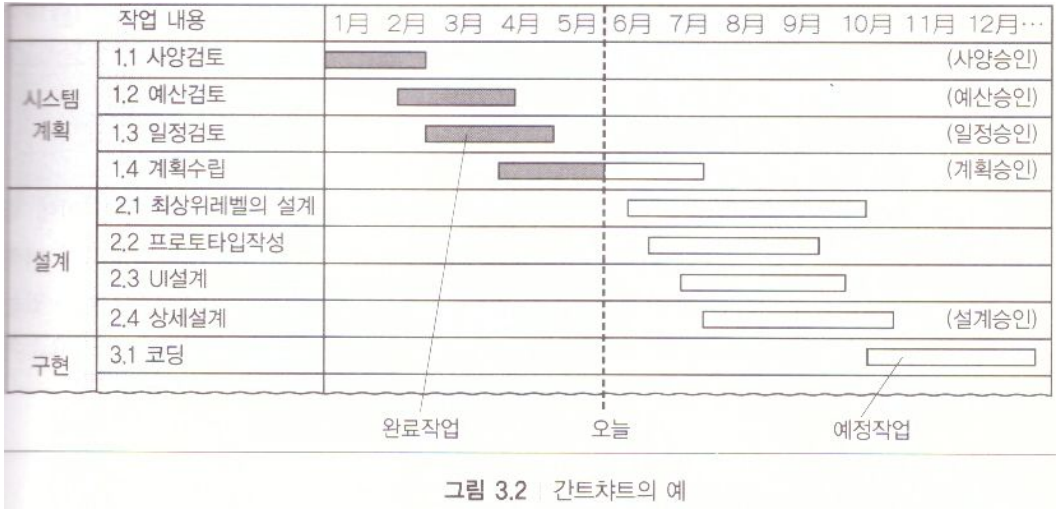


그림 3.2 간트챠트의 예

- 검은색으로 색칠되어 있는 막대그래프(직사각형)는 오늘(6월1일)까지 작업이 완료되어 있음을 나타냄
  - "1.4 계획수립"에 관해서는 현재작업중이며, 7월하순까지 계속될 예정
  - "2.1 최상위 레벨의 설계"에 관해서는 6월중순부터 작업을 시작할 예정
  - 마일스톤(milestone) : 개발일정에 있어서 작업완료시점(특정한 작업의 종료시점)

3.1.2 개발계획

- (7) 개발체제
  - 개발공수의 산정과 일정에 맞춘 프로젝트조직의 활동, 필요한 개발인력의 할당 등의 개발체제
  - 프로젝트조직은 몇 개의 팀들로 구성되며, 각 팀이 책임을 지고 담당 작업을 수행하게 됨
  - 팀내의 운영뿐만아니라 팀들간의 의사소통이 중요
    - 팀들 사이에서 공유되는 정보와 공유하기 위한 방법을 결정

- (8) 개발환경 및 개발방법
  - 소프트웨어를 개발하기 위해 필요한 환경과 도구
  - 하드웨어환경
    - 문서를 작성 및 관람하기 위한 컴퓨터
    - 프로그래밍작업을 수행하기 위한 컴퓨터
    - 테스트 작업에 사용될 컴퓨터
  - 소프트웨어환경
    - OS, 프로그래밍언어, 컴파일러, 각종 개발도구, 의사소통을 위한 도구 등
  - 개발에서 사용되는 방법론과 코딩규약, 테스트 방침 등 결정

3.1.2 개발계획

- (9) 성과물의 관리방법
  - 각 개발공정에서 작성된 성과물들을 적절하게 관리하기 위한 방법
    - 이와같은 작업을 “소프트웨어구성관리”라고 부름
  - 성과물을 단순하게 기록해두는 것뿐만아니라, 누가 언제 작성했는지, 개발도중에 어떠한 변경이 일어났는지, 성과물들 사이의 관계 등에 대해서도 관리

- (10) 위험요소의 관리방법
  - 소프트웨어개발을 진행하는데 있어서 발생이 예측되는 위험요소들과 이에 대한 대처방법, 발생한 위험요소에 대한 대책뿐만아니라, 위험요소가 발생하기 어려운 환경을 만드는 것도 위험요소관리에 포함됨
    - 위험요소충격 : 위험요소에 의해 발생될 수 있는 손실
    - 위험요소확률 : 위험요소가 발생할 가능성
    - 위험요소통제 : 위험요소의 영향을 최소화하거나 회피하는 것

3.2 개발공수의 산정

개요

- 프로젝트계획을 수립하는데 있어서 개발하려는 소프트웨어에 관한 비용을 산정하는 것은 필수
- 고객의 입장에서는 해당 소프트웨어개발에 어느정도의 비용과 기간이 필요한지 알고싶어함
- 개발공수를 산정하거나, 이를 토대로 개발비용을 산출
    - 개발공수 산출 기법: 표준업무기법, COCOMO, 기능점수법 등

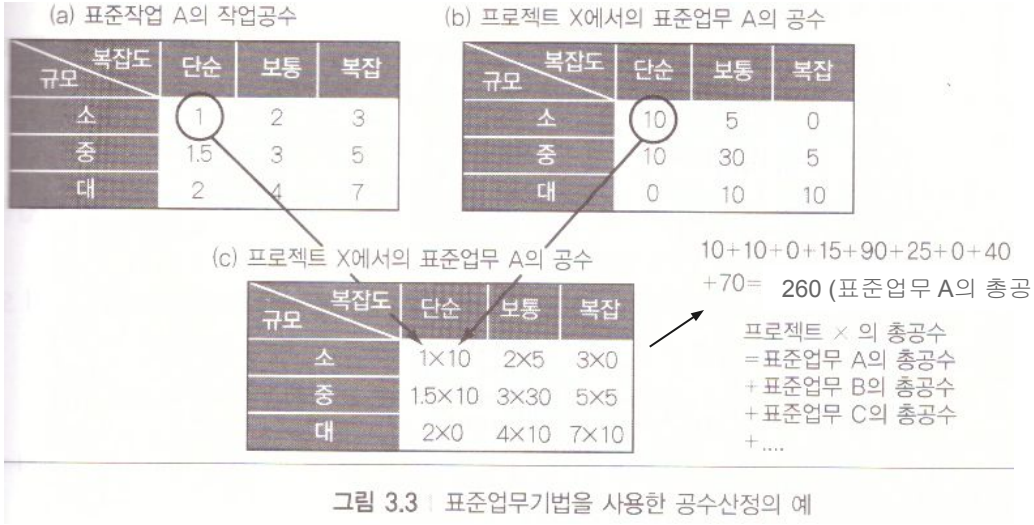
목차

3.2 개발공수의 산정

선문대학교

3.2.1 표준업무기법

- 표준업무기법
  - 소프트웨어개발에 필요한 작업("표준업무")마다 개발공수를 미리 설정하고
    이를 토대로 실제 소프트웨어개발에 필요한 작업을 예측하여, 이에따른 공수를 산정하는 산출방법
- 작업공수는 작업의 규모와 복잡도 매트릭스(행렬)에 설정되어 있음
  - 그림(a): 표준업무A의 "소규모"-"단순" 작업의 경우, 해당 공수는 1(즉, 1일)
  - 매트릭스는 각 기업 및 개발프로젝트의 실적과 멤버들의 경험 등을 토대로 독자적으로 설정



규모 \ 복잡도	단순	보통	복잡
소	1	2	3
중	1.5	3	5
대	2	4	7

규모 \ 복잡도	단순	보통	복잡
소	10	5	0
중	10	30	5
대	0	10	10

규모 \ 복잡도	단순	보통	복잡
소	1×10	2×5	3×0
중	1.5×10	3×30	5×5
대	2×0	4×10	7×10

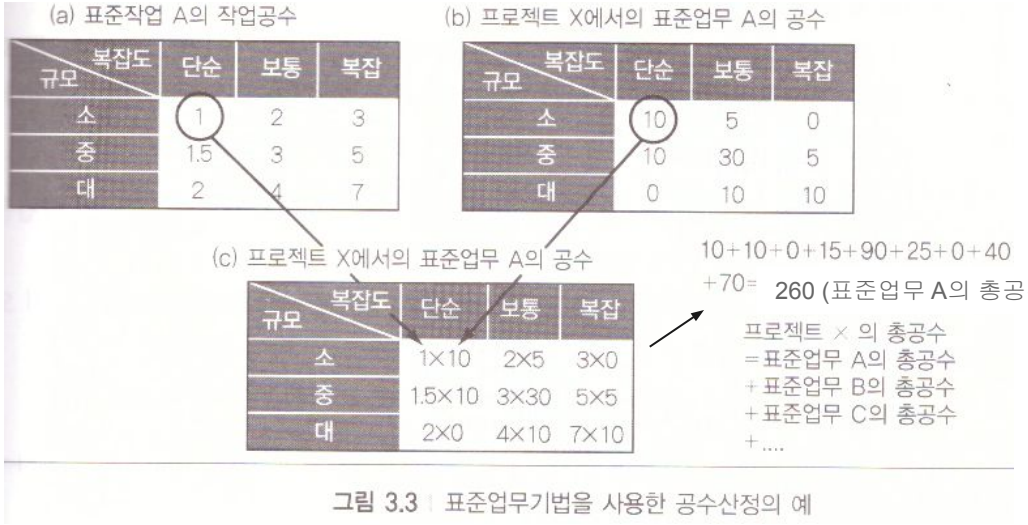
그림 3.3 | 표준업무기법을 사용한 공수산정의 예

3.2 개발공수의 산정

선문대학교

3.2.1 표준업무기법

- 공수 산정: 개발하려는 소프트웨어의 작업을 여러 개의 표준업무들로 분해하고
  각 표준업무들의 규모와 복잡도를 결정(예측)
- 각 종류별 표준업무의 작업공수와 건수를 곱하여 합산함으로써 해당 표준업무의 총공수를 산출
  - 그림(b): 특정한 프로젝트X에 대해서 표준업무A가 몇 번 실시되는지에 관한 건수를 나타낸 매트릭스, 표준업무A의 총공수는 260일
  - 모든 표준업무에 관해서 공수를 산출하고 합계함으로써 프로젝트 전체의 공수를 산정할 수 있음



규모 \ 복잡도	단순	보통	복잡
소	1	2	3
중	1.5	3	5
대	2	4	7

규모 \ 복잡도	단순	보통	복잡
소	10	5	0
중	10	30	5
대	0	10	10

규모 \ 복잡도	단순	보통	복잡
소	1×10	2×5	3×0
중	1.5×10	3×30	5×5
대	2×0	4×10	7×10

그림 3.3 | 표준업무기법을 사용한 공수산정의 예

3.2.2 COCOMO(COnstructive COst MOdel)

- COCOMO
  - Boehm에 의해 제안
  - 소프트웨어 개발규모와 개발공수의 관계를 통계적인 모델로부터 추측하는 기법
  - 개발하려는 소프트웨어의 규모를 주석문을 제외시킨 소스코드의 라인수(LOC : Line Of Code)를 사용하여 측정
  - 소스코드의 라인수를 변환모델에 입력함으로써 개발공수가 출력됨
  - 장점 : 개발규모만을 사용하여 통계적으로 개발공수를 산출할 수 있음

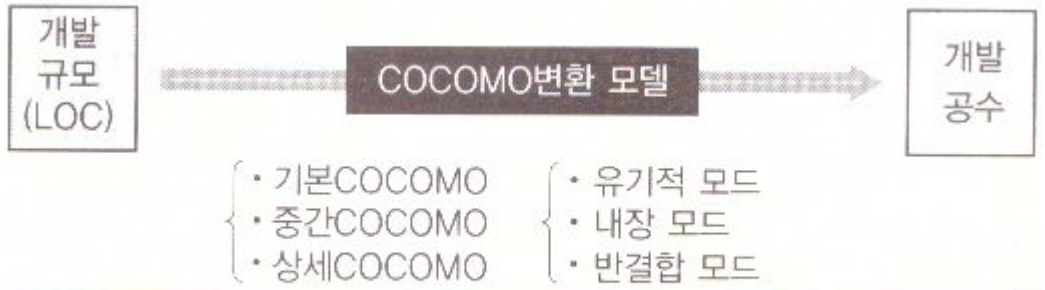


그림 3.4 | COCOMO에 의한 개발 공수의 산정

3.2 개발공수의 산정

3.2.2 COCOMO

- 산정작업을 실시하는 시기에 따른 3가지 방법

- (1) 기본COCOMO
  - 개발규모만으로 개발공수를 산출
  - 프로젝트의 계획단계에서 사용

- (2) 중간COCOMO
  - 개발하려는 소프트웨어의 특징과 멤버들의 경험 및 능력에 따라서 개발규모를 조정하여 개발공수를 산출
  - 주로 요구사항정의가 완료된 후에 사용

- (3) 상세COCOMO
  - 모듈구성 등을 고려하여 개발규모를 조정하여 개발공수를 산출
  - 주로 설계종료 후에 사용

3.2.2 COCOMO

- 소프트웨어개발의 종류에 따른 3가지 모드

- (1) 유기적 모드
  - 소수의 개발인력으로 수행하는 단순하고 소규모적인 개발
  - 개발전체를 자사에서 수행하고 있는 경우에 해당

- (2) 내장 모드
  - 복잡하고 대규모 또는 엄격한 제약사항을 갖는 개발

- (3) 반결합 모드
  - 일반적인 업무처리소프트웨어의 개발
  - 유기적 모드와 내장 모드의 중간에 해당

3.2 개발공수의 산정

3.2.2 COCOMO

- 각 모드에는 변환모델이 있으므로 적절한 변환모델을 사용하여 개발공수를 산출하면 됨

- 인월값(MM: Man-Month값)
  - 최종적인 개발공수
  - 한 사람이 1개월간 소프트웨어개발작업에 투입하는 시간
    - 예: 10 MM은 2명이 5개월간 투입하는 시간 또는 5명이 2개월간 투입하는 시간 등
    - 단, 실제 개발에 있어서 인간과 시간은 치환가능한 것이 아니므로, 2명을 5개월간 투입해야하는 작업에 대해서, 5명을 투입하였다고 해서 2개월만에 작업이 종료되는 것이 아니라는 점에 주의

3.2 개발공수의 산정

3.2.2 COCOMO

- 개발규모로부터 개발공수를 산정하는 모델
  - COCOMO
    - 중~대규모(수만~수천라인)의 소프트웨어에 적합한 모델
  - Doty모델
    - 소규모(수만라인 이하)의 소프트웨어에 적합한 모델
  - Putnam모델
    - 수십만 라인이상의 초대형규모의 소프트웨어에 적합한 모델

3.2.3 COCOMO II

- COCOMO II 의 제안 배경
  - 초기 COCOMO
    - 소프트웨어의 규모 (예: 소스코드의 라인수) 를 예측할 수 있음을 전제
    - 그러나, 소프트웨어개발의 분석공정 및 설계공정
      - 소프트웨어의 규모 추정이 곤란한 경우, 여러가지 문제점 발생
        - 산정 불가능
        - 산출한 결과값의 부정확성
  - 개발공정 초기에 소스코드의 라인수를 파악하는 것은 어려움이 있다는 사실을 감안하게 됨

3.2.3 COCOMO II

- COCOMO II에서는 3가지 모델을 사용하여 개발공수를 산정
  - (1) 어플리케이션 조립모델(Application Composition Model)
    - GUI빌더 또는 컴포넌트를 사용한 프로토타입개발에서 객체점수(object points)의 관점에서 소프트웨어의 규모를 산정하는 모델
      - 객체점수: 시스템내에 등장하는 객체(화면 및 장표 등)에 의해 제공되는 기능의 복잡도를 단순/중간/곤란으로 분류하고, 각각의 등장횟수에 가중치를 부여하여 산출한 결과값
  - (2) 초기설계모델(Early Design Model)
    - 기능점수법을 사용하여, 기능의 갯수로부터 규모를 측정하는 모델
    - 설계초기단계에서 전체 시스템의 구조가 명확하게 정의되기 전에 사용
  - (3) 구조설계이후모델(Post-Architecture Model)
    - 기능점수법과 측정된 소스코드의 라인수에 의해 개발공수를 산출하는 모델
    - 외부설계에 의해 시스템구조가 정의된 후에 사용
    - 이 시기에는 개발소프트웨어에 관한 많은 정보를 얻을 수 있기 때문에, 다양한 종류의 비용계수를 도입하고 있음

3.2.4 기능점수법(FP법)

- 객체지향소프트웨어 개발 증가
  - 라인수를 사용하여 소프트웨어의 규모를 나타내지 않는 객체지향소프트웨어 개발이 증가하고 있음

- 기능점수법(FP법)
  - Allan J. Albrecht에 의해 제안
  - 소스코드의 라인수를 대신할 수 있는 것으로서 소프트웨어에 포함된 기능의 개수(Function Point)에 의해 소프트웨어의 규모를 산출하는 방법

3.2.4 기능점수법

- 소프트웨어의 내부에서 어떤 처리가 수행되고 있는지(또는 수행될 예정인지)를 분석하여 5가지 기능으로 분류

- (1) 외부입력(EI: External Inputs)
  - 외부로부터 소프트웨어로의 입력
  - 소프트웨어의 내부논리파일에 대한 갱신이 수반됨
  - 파일의 형식 및 파일에 대한 처리내용이 서로 다른 경우에는 각각 별개로 카운트

- (2) 외부출력(EO: External Outputs)
  - 소프트웨어로부터 외부(이용자 등)로의 출력
  - 외부입력과 마찬가지로, 파일의 형식 및 파일에 대한 처리내용이 서로 다른 경우에는 각각 별개로 카운트

3.2.4 기능점수법

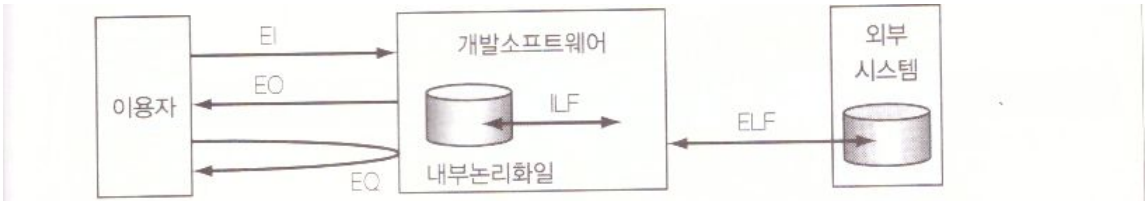
- (3) 외부조회 (EQ: External inQuiries)
  - 외부로부터 소프트웨어로의 조회
    - 조회: 입출력에 의해 소프트웨어의 내부논리파일이 갱신되지 않는 것을 말함

- (4) 내부논리파일 (ILF: Internal Logical Files)
  - 소프트웨어내부에 존재하는 파일의 규모
  - 파일의 개수를 카운트하는 것이 아니라, 데이터로서 의미를 갖는 그룹(논리적 레코드)의 개수를 카운트

- (5) 외부논리파일 (ELF: External Logical Files)
  - 다른 시스템에서 관리되어 소프트웨어가 참조하는 파일의 규모
  - 내부논리파일처럼 논리적인 레코드수를 카운트

3.2.4 기능점수법

- 복잡도 매트릭스를 토대로, 시스템내부의 각 기능들을 단순/보통/복잡으로 구분
  - 복잡도는 각 기능에 관련된 파일 및 파일내부의 레코드수와 각 기능에서 처리하는 데이터항목수에 의해 결정됨
  - 그림 예: 외부입력(EI)에 있어서 관련파일수가 2인 경우, 다루는 데이터항목수가 1~4인 경우에는 "단순", 5~15인 경우에는 "보통", 16이상인 경우에는 "복잡"으로 정의하고 있음



EI의 복잡도

관련 화일수	데이터 항목수		
	1~4	5~15	16~
0~1	단순	단순	보통
2	단순	보통	복잡
3~	보통	복잡	복잡

EO, EQ의 복잡도

관련 화일수	데이터 항목수		
	1~5	6~19	20~
0~1	단순	단순	보통
2~3	단순	보통	복잡
4~	보통	복잡	복잡

ILF, ELF의 복잡도

레코드 수	데이터 항목수		
	1~19	20~50	51~
0~1	단순	단순	보통
2~5	단순	보통	복잡
6~	보통	복잡	복잡

그림 3.5 기능점수법에서의 기능분류와 복잡도에 관한 메트릭스

3.2.4 기능점수법

- 이와같은 정의에 따라서 기능수를 카운트함으로써, 그림(a)와 같은 5가지 기능분류와 3가지 복잡도에 관한 기능수 매트릭스를 작성할 수 있음

(a) 복잡도 별 기능수

기능 \ 복잡도	단순	보통	복잡
EI	10	9	15
EO	14	12	14
EQ	2	4	7
ILF	3	8	6
EIF	4	5	5

(b) 가중치 계수

기능 \ 복잡도	단순	보통	복잡
EI	×3	×4	×6
EO	×4	×5	×7
EQ	×3	×4	×6
ILF	×7	×10	×15
EIF	×5	×7	×10

(c) 기능점수

기능 \ 복잡도	단순	보통	복잡
EI	10×3	9×4	15×6
EO	14×4	12×5	14×7
EQ	2×3	4×4	7×6
ILF	3×7	3×10	6×15
EIF	4×5	5×7	5×10

30+36+90
+56+60+98
+ 6+16+42
+21+30+90
+20+35+50
=680 (미조정FP)

그림 3.6 기능점수의 산출

3.2.4 기능점수법

- 계수(가중치)를 미리 부여
  - 5가지 기능분류 및 3가지 복잡도 고려
- 전체적인 기능점수
  - 카운트된 기능수와 계수를 곱셈 결과를 합계
  - 그림(b)의 가중치계수가 주어진 경우, 외부입력(EI)의 값: 예) 그림(c) 10X3=30
- 개발소프트웨어의 기능점수
  - 모든 기능분류 및 복잡도에 대해서 같은 방식으로 계산, 결과들을 합계한 수치: 예) 그림(c) 680
  - 단, 이러한 수치는 최종결과는 아니며, "미조정 기능점수"라고 부름

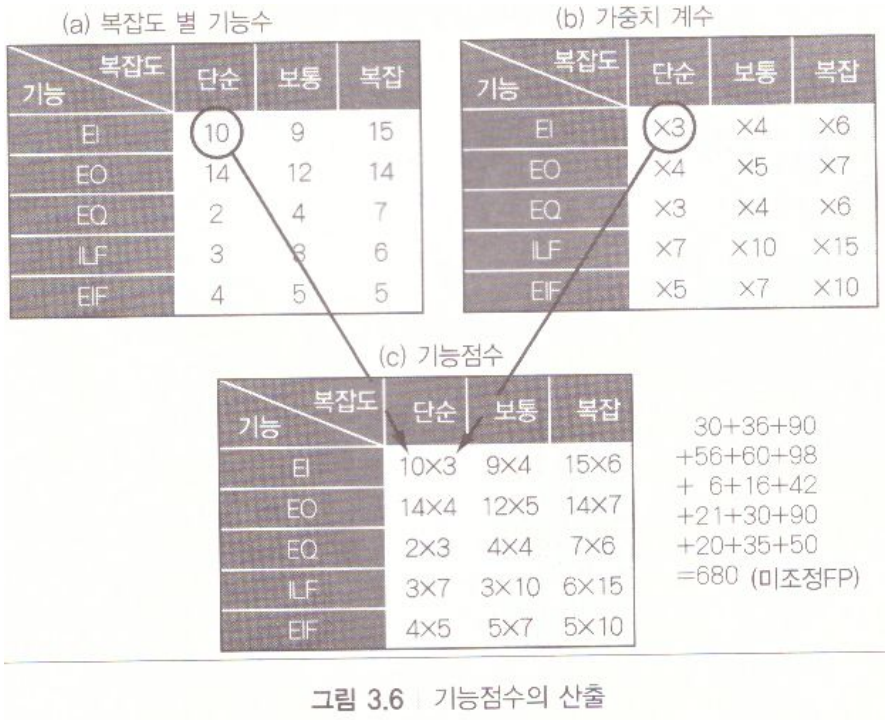


그림 3.6 | 기능점수의 산출

3.2.4 기능점수법

- 미조정기능점수 조정 필요
  - 소프트웨어의 특성에 따라 최종적인 기능점수를 산출하기 위해
- 소프트웨어의 특성에 대한 평가 항목
  - 예: 데이터통신, 분산처리, 성능 등 항목 14개
  - 각 항목에 대해 아래 6단계로 점수 부여
    - 0: 전혀 관계없다.
    - 1: 거의 영향을 받지 않는다.
    - 2: 적당하게 영향을 받는다.
    - 3: 평균적인 영향을 받는다.
    - 4: 큰 영향을 받는다.
    - 5: 매우 커다란 영향을 받는다.

3.2 개발공수의 산정

3.2.4 기능점수법

- 소프트웨어의 특성에 관한 합계점수가 P 일때,
  - 조정용 계수 C = 0.65 + (0.01 X 합계점수P)
    - 0.65와 0.01은 상수이며, 합계점수P의 최대값은 70(14항목 X 5점)

- 조정후의 기능점수FP를 구하는 공식
  - FP = (조정용 계수C) X (미조정기능점수)
    - 조정계수C의 범위는 0.65~1.35가 되므로, ±35%의 조정을 하는 것이 됨
    - 예: 소프트웨어의 특성에 관한 합계점수P가 28인 경우,
      조정용 계수C는 0.93(=0.65 + 0.01 X 28)이 되어,
      최종적인 기능점수FP는 632.4(=0.93 X 680)

- 기능점수는 개발규모를 표현하고 있을 뿐, 개발공수를 직접 나타내는 것은 아님
- 기능점수로부터 개발공수로의 변환모델은 각 기업 및 프로젝트가 경험과 실적을 토대로 결정할 필요가 있음

3.3 품질관리

개요

- 프로젝트관리에 있어서 품질 평가 필수
  - 소프트웨어개발 성과물(Products)의 품질, 프로세스의 품질
- 소프트웨어의 품질 평가 지침
  - 품질특성, 소프트웨어매트릭스
- 개발능력성숙도모델(CMM)
  - 개발프로세스의 평가와 개선 목적

목차

3.3.1 소프트웨어의 품질특성

- 소프트웨어의 품질에 관한 국제표준규격 ISO/IEC9126 은 6가지 품질특성과 부특성으로 구성

표 3.2 소프트웨어의 품질특성(ISO/IEC 9126)

품질특성	설명	부특성
기능성	필요한 기능이 구현되어 있는가	합목적성, 정확성, 상호운용성, 보안, 표준적합성
신뢰성	기능이 정상적으로 계속 동작하는가	성숙성, 장해허용성, 회복성, 표준적합성
사용성	사용자가 사용하기 수월한가	이해성, 습득성, 조작성, 매력성, 표준적합성
효율성	목적달성을 위해 사용하는 자원이 적절한가	시간적 효율성, 자원효율성, 표준적합성
유지 보수성	개정작업에 필요한 노력은 적은가	해석성, 변경성, 안정성, 시험성, 표준적합성
이식성	다른 환경으로 이식하기 수월한가	순응성, 설치성, 공존성, 치환성, 규격준거성

3.3.1 소프트웨어의 품질특성

- (1) 기능성
  - 합목적성: 소프트웨어가 사용자의 목적에 합치되는 기능을 제공하고 있는가
  - 정확성: 소프트웨어가 사양에 대해서 올바르게 동작하는가
  - 상호운용성: 소프트웨어가 지정된 다른 시스템과 서로 운용가능한가
  - 보안: 소프트웨어에 대한 부당한 액세스를 배제시킬 수 있는가
  - 표준적합성: 소프트웨어의 기능이 법률, 업계표준, 규격을 준수하고 있는가

- (2) 신뢰성
  - 성숙성: 장애가 발생했을 경우에 소프트웨어가 정지하지 않는가
  - 장애허용성: 장애가 발생하더라도 소프트웨어가 기능을 계속 제공할 수 있는가
  - 회복성: 장애가 발생한 후에 신속하게 소프트웨어의 기능이 정상으로 복귀하는가
  - 표준적합성: 소프트웨어의 신뢰성이 법률, 업계표준, 규격을 준수하고 있는가

3.3.1 소프트웨어의 품질특성

- (3) 사용성
  - 이해성: 소프트웨어의 사용법을 이용자가 이해하기 수월한가
  - 습득성: 처음 사용하는 사용자도 곧바로 소프트웨어를 사용할 수 있는가
  - 조작성: 이용자가 소프트웨어를 사용할 때의 유저인터페이스가 사용하기 수월한가
  - 매력성: 소프트웨어가 이용자에게 매력적인가
  - 표준적합성: 소프트웨어의 사용성이 법률, 업계표준, 규격을 준수하고 있는가

- (4) 효율성
  - 시간적 효율성: 응답시간이 짧은가. 처리속도가 빠른가. 지정된 실행경로가 확보가능한가
  - 자원효율성: 메모리와 네트워크 등의 자원을 쓸모없이 낭비하고 있지않나
  - 표준적합성: 소프트웨어의 성능이 법률, 업계표준, 규격을 준수하고 있는가

3.3.1 소프트웨어의 품질특성

- (5) 유지보수성
  - 해석성: 소프트웨어 장애가 발생했을때, 수정이 필요한 부분을 지정하기 수월한가
  - 변경성: 변경요구에 대하여 소프트웨어의 변경작업이 수월한가
  - 안정성 : 소프트웨어를 수정한 경우, 예상밖에 있는 부분에 영향을 미치는 경우는 없는가
  - 시험성: 소프트웨어를 수정한 경우, 테스트하기 수월한가
  - 표준적합성: 소프트웨어의 유지보수성이 법률, 업계표준, 규격을 준수하고 있는가

- (6) 이식성
  - 순응성: 소프트웨어를 다른 환경으로 이동시키는 경우의 수고가 적은가
  - 설치성: 지정된 환경에 소프트웨어를 설치하기 수월한가
  - 공존성: 소프트웨어가 같은 환경에서 다른 소프트웨어와 공존가능한가
  - 치환성: 같은 목적을 갖는 다른 소프트웨어로 바꾸는 것은 가능한가
  - 규격준거성: 소프트웨어의 이식성이 법률, 업계표준, 규격을 준수하고 있는가

3.3 품질관리

3.3.1 소프트웨어의 품질특성

- 소프트웨어의 품질을 향상시키기 위해서는
  - 각각의 품질특성에 관해서 높은 수준을 유지하는 것이 중요
    - 그러나, 개발비용 및 일정이 유한하기 때문에 어려움
- 프로젝트관리에서는
  - 개발소프트웨어가 만족해야되는 품질특성에 우선순위를 부여하여 품질을 관리하는 것이 좋음

3.3.2 소프트웨어매트릭스

- 프로젝트관리에서 산출물과 프로세스를 평가하는 경우에는 정량적인 평가척도가 도움이 됨. 이러한 평가척도를 매트릭스라고 부르며, 다음과 같은 2가지가 있음

- (1) 산출물 매트릭스
  - 소프트웨어개발에서 작성된 성과물을 정량적으로 평가하는 척도
    - 예: 소프트웨어의 규모를 측정하는 매트릭스로서, 소스코드의 라인수와 프로그램의 명령수(스텝수)가 있음
  - Halstead 척도
    - 단순히 라인수와 명령수를 카운트하는 것이 아니라, 소스코드에 포함된 연산자와 피연산자의 종류수 및 출현횟수로부터 프로그래밍 노력을 평가하는 방법

3.3.2 소프트웨어매트릭스

- 프로젝트관리에서 산출물과 프로세스를 평가하는 경우에는 정량적인 평가척도가 도움이 됨. 이러한 평가척도를 매트릭스라고 부르며, 다음과 같은 2가지가 있음

- (1) 산출물 매트릭스
  - McCabe의 순환복잡도
    - 프로그램의 복잡도를 평가하는 척도
    - 프로그램의 실행흐름(제어흐름)에서 일차독립적인 경로의 개수를 사용하여 복잡도를 측정하는 기법
    - 예: 프로그램에 조건분기문이나 반복문이 존재하지 않는 경우, 프로그램의 실행경로는 1이기 때문에, 복잡도는 1이 됨. 조건분기문이 1개만 존재하는 경우에는, 1차독립적인 경로는 참과 거짓의 경우로 나누어지므로 2가 되며, 복잡도는 2가 됨
  - CK매트릭스(Chidamber and Kemerer's metrics)
    - 객체지향 소프트웨어에서 복잡도 측정에 특화된 매트릭스

3.3.2 소프트웨어매트릭스

- (2) 프로세스매트릭스
  - 성과물을 작성하는 작업의 효율을 평가하는 척도
  - 단순한 방법으로서 특정한 작업에 투입된 시간과 자원의 양을 측정
  - 특정한 프로세스를 실행중에 발생하는 이벤트의 개수를 측정함으로써 해당 작업의 효율을 평가하는 것도 가능
  - 이벤트의 예
    - 코드검사(완성된 소스코드가 규정대로 기술되어 있는지를 확인하는 작업)중에 발견된 오류의 개수
    - 요구사항변경에 대한 수정코드의 라인수

3.3.2 소프트웨어매트릭스

- 품질평가에 매트릭스를 사용하는 경우에 곤란한 점은 적절한 매트릭스의 선택임
- GQM(Goal-Question-Metric)
  - Victor R. Basili 등에 의해 제안
  - 측정데이터를 수집하는 목적, 목적을 달성하기 위해 파악하려는 정보를 표현한 질문, 질문에 대한 정량적인 응답을 얻기위한 매트릭스를 각각 Goal층, Question층, Metrics층으로 나누어서 톱다운방식으로 기술하고, 목적과 질문, 질문과 매트릭스 사이의 관계를 정의
  - GQM을 사용함으로써 선택한 매트릭스에 대한 측정 목적과 측정 데이터의 선택이 명확하게 됨

3.3.3 프로세스의 평가와 개선

- CMM(Capability Maturity Model)
  - 미국 카네기멜론대학 소프트웨어공학연구소 (SEI: Software Engineering Institute)에서 개발
  - 프로세스의 개선지원을 목적으로 하여, 조직의 개발능력에 대한 성숙도를 평가하는 기법
  - 현재는 CMMI(CMM Integration)로서 체계화
  - 소프트웨어 CMM(SW-CMM) : CMMI에서 소프트웨어개발에 관련있는 것
    - SW-CMM에서는 성숙도 수준을 5단계로 규정하고, 각 수준에 따라서 소프트웨어를 개선하는 절차를 나타냄
    - SW-CMM을 토대로 프로세스를 개선함으로써 프로젝트관리 수준을 향상시키는 것을 기대할 수 있음

3.3.3 프로세스의 평가와 개선

- SW-CMM 성숙도
  - 레벨2(반복가능) : 어떤 조직에서 기본적인 관리공정이 존재하고, 과거에 성공했던 공정을 반복가능
  - 레벨3(기정의) : 조직이 개발공정을 정의함으로써 개발공정에 대한 문서화 및 표준화가 실시되어 과거에 성공했던 것 뿐만아니라 모든 공정에서 표준공정을 이용가능
  - 레벨4(관리화) : 조직이 개발공정의 정량적 관리를 수행함으로써 개발공정과 산출물의 정량적 데이터를 수집하고, 데이터분석과 데이터를 토대로 공정관리가 가능

- CMMI 외에도, 소프트웨어제품에 대한 품질보증을 규정한 ISO/IEC 9000-3이 있음

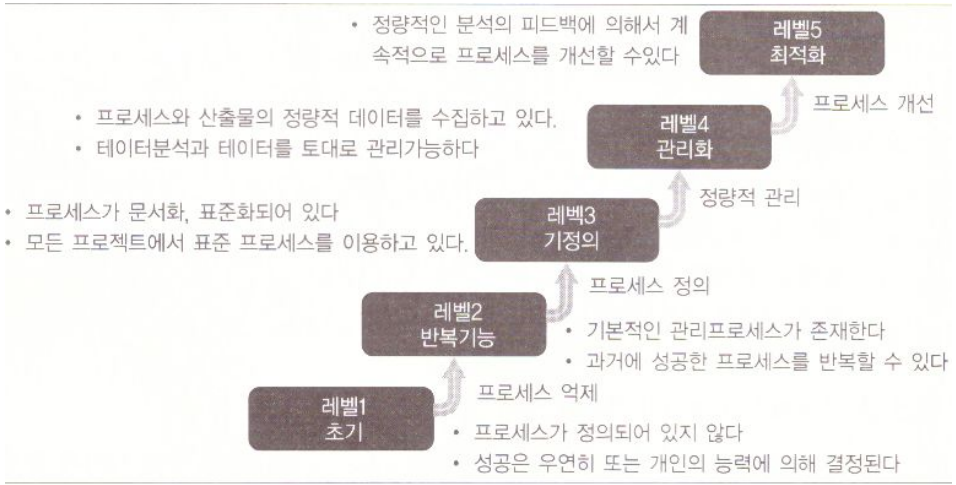


그림 3.7 SW-CMM에서의 성숙도

연습문제 (객관식)

각 질문에 대한 알맞은 답의 번호(숫자)를 괄호안에 표시하시오.

❑ 1. 프로젝트관리지식체계 (Project Management Body Of Knowledge : PMBOK)에서 프로젝트의 프로세스들을 5가지 그룹으로 분류하고 있다. 그 중 프로젝트계획과의 차이점을 인식하기 위하여 실적보고 및 진척상황에 대한 측정을 수행하고, 변경관리 및 위험요소에 대한 대응에 관한 활동을 수행하는 프로세스는 무엇인가? ( )
(1) 착수프로세스 (2) 계획프로세스 (3) 실행프로세스
(4) 통제프로세스 (5) 종료프로세스

❑ 2. 소프트웨어개발을 진행하는데 있어서 발생이 예측되는 위험요소들과 이에 대한 대처방법, 발생한 위험요소에 대한 대책뿐만아니라, 위험요소가 발생하기 어려운 환경을 만드는 것도 위험요소관리에 포함된다. 위험요소의 영향을 최소화하거나 회피하는 것을 무엇이라고 하는가? ( )
(1) 위험요소대책 (2) 위험요소충격 (3) 위험요소통제
(4) 위험요소확률 (5) 위험요소종료

객관식 문제 (답안지)

❑ 1. (4) → 통제프로세스

❑ 2. (3) → 위험요소통제

연습문제 (객관식)

각 질문에 대한 알맞은 답의 번호(숫자)를 괄호안에 표시하시오.

- 3. Boehm에 의해 제안된 개발공수를 산출하는 기법으로서, 개발하려는 소프트웨어의 규모를 주석문을 제외시킨 소스코드의 라인수(LOC : Line Of Code)를 사용하여 측정하는 것은 무엇인가? (　　)
  (1) 표준업무기법 (2) COCOMO (3) 기능점수법
  (4)프로젝트관리지식체계 (5) 작업명세구조

- 4. COCOMO의 3가지 모드 중 해당하지 않는 것 2가지를 고르시오. (　　)
  (1) 외장 모드 (2) 유기적 모드 (3) 내장 모드 (4) 결합모드 (5) 반결합모드

- 5. COCOMO II의 모델 중 GUI빌더 또는 컴포넌트를 사용한 프로토타입개발에서 객체점수(object points)의 관점에서 소프트웨어의 규모를 산정하는 모델은 무엇인가? (　　)
  (1) 어플리케이션 조립모델 (2) 초기설계모델
  (3) 구조설계이후모델 (4) 중간설계모델 (5) 평가설계모델

- 6. 소프트웨어의 6가지 품질특성에 속하지 않는 것을 고르시오. (　　)
  (1) 기능성 (2) 신뢰성 (3) 평가성 (4)효율성 (5) 유지보수성

객관식 문제 (답안지)

- 3. (2) → COCOMO
- 4. (1, 4) → COCOMO의 3가지 모드: 유기적, 내장, 반결합모드
- 5. (1) → 어플리케이션 조립모델
- 6. (3) → 6가지 품질속성: 기능성, 신뢰성, 사용성, 효율성, 유지보수성, 이식성

연습문제 (주관식)

선문대학교

주관식 문제 (괄호안에 알맞은 단어를 입력하시오.)

- 1. (　　)란 요구사항을 만족하는 소프트웨어를 고객에게 정해진 기간내에 납품하기 위해서, 소프트웨어개발에서 누가 어떤 단계에서 무엇을 하면 좋은지를 계획하는 작업이다. 동시에, 계획한대로 프로젝트가 진행되고 있는지를 확인하고, 그 결과에 따른 대책작업도 포함된다.

- 2. 프로젝트관리지식체계 (Project Management Body Of Knowledge : PMBOK)에서 프로젝트의 프로세스들을 5가지 그룹으로 분류하고 있다. 무엇인지 빈 칸을 채우시오.
(　　)프로세스 → (　　)프로세스 → (　　)프로세스 → (　　) 프로세스 → (　　)프로세스

- 3. 개발일정을 기술하는 도구로서, (　　)는 프로젝트에서 실행되는 작업(Activity)들을 분할하고 상세화해서 기술한 것이고, (　　)는 각 작업들을 병렬로 기술하고, 각각의 작업기간을 막대그래프형태로 나타낸 것이다.

주관식 문제 (답안지)

- 1. 프로젝트관리
- 2. 착수 - 계획 - 실행 - 통제 - 종료
- 3. 작업명세구조 (Work Breakdown Structure : WBS), 간트차트 (Gantt Chart)

연습문제 (주관식)

선문대학교

주관식 문제 (괄호안에 알맞은 단어를 입력하시오.)

- 4. 개발일정에 있어서 작업완료시점(특정한 작업의 종료시점)을 (　　)이라고 부른다.
- 5. (　　)은 소프트웨어개발에 필요한 작업(　　)마다 개발공수를 미리 설정하고 이를 토대로 실제 소프트웨어개발에 필요한 작업을 예측하여, 이에따른 공수를 산정하는 산출방법이다.
- 6. (　　)란 한 사람이 1개월간 소프트웨어개발작업에 투입하는 시간을 의미한다.
- 7. (　　)란, 시스템내에 등장하는 객체(화면 및 장표 등)에 의해 제공되는 기능의 복잡도를 단순/중간/곤란으로 분류하고, 각각의 등장횟수에 가중치를 부여하여 산출한 결과값이다.

주관식 문제 (답안지)

- 4. 마일스톤(milestone)
- 5. 표준업무기법, 표준업무
- 6. 인월값 또는 Man-Month
- 7. 객체점수

연습문제 (주관식)

주관식 문제 (괄호안에 알맞은 단어를 입력하시오.)

- 8. Allan J. Albrecht에 의해 제안한 (　　)은 소스코드의 라인수를 대신할 수 있는 것으로서, 소프트웨어에 포함된 기능의 개수(Function Point)에 의해 소프트웨어의 규모를 산출하는 방법이다.

- 9. (　　)란 소프트웨어를 정량적으로 평가하는 척도를 말한다. (　　)는 소프트웨어개발에서 작성된 성과물을 정량적으로 평가하는 척도이고, (　　)는 성과물을 작성하는 작업의 효율을 평가하는 척도이다.

주관식 문제 (답안지)

- 8. 기능점수법 (FP법)

- 9. 매트릭스, 산출물 매트릭스, 프로세스매트릭스

연습문제 (주관식)

주관식 문제 (괄호안에 알맞은 단어를 입력하시오.)

- 10. GQM기법에서는 측정데이터를 수집하는 목적, 목적을 달성하기 위해 파악하려는 정보를 표현한 질문, 질문에 대한 정량적인 응답을 얻기위한 매트릭스를 각각 (　　)층, (　　)층, (　　)층으로 나누어서 톱다운방식으로 기술하고, 목적과 질문, 질문과 매트릭스 사이의 관계를 정의한다.

- 11. 미국 카네기멜론대학 소프트웨어공학연구소 (SEI: Software Engineering Institute)에서 개발된 (　　)은 프로세스의 개선지원을 목적으로 하여, 조직의 개발능력에 대한 성숙도를 평가하는 기법이다.

주관식 문제 (답안지)

- 10. Goal, Question, Metrics

- 11. CMM(Capability Maturity Model)

Any questions?