5과목 〈정보시스템 구축 관리〉

1장 소프트웨어 개발 방법론 활용

섹션 166 소프트웨어 개발 방법론

: 소프트웨어 개발, 유지보수 등에 필요한 여러 가지 일들의 수행 방법과 이러한 일들을 효율적으로 수행하려는 과정에서 필요한 각종 기법 및 도구를 체계적으로 정리하여 표준화한 것

- 소프트웨어 개발 방법론의 목적은 소프트웨어의 생산성과 품질 향상이다
- 소프트웨어 개발 방법론의 종류에는 구조적 방법론, 정보공학 방법론, 객체지향 방법론, 컴포넌트 기반(CBD) 방법론, 애자일 방법론, 제품 계열 방법론 등이 있다

1. 구조적 방법론 : 정형화된 분석 절차에 따라 사용자 요구사항을 파악하여 문서화하는 처리 중심의 방법론
   - 1960년대까지 가장 많이 적용되었던 소프트웨어 개발 방법론
   - 쉬운 이해 및 검증이 가능한 프로그램 코드를 생성하는 것이 목적
   - 복잡한 문제를 다루기 위해 분할과 정복(Devide and Conquer) 원리 적용
   - 구조적 방법론의 절차
   : 타당성 검토 단계 → 계획 단계 → 요구사항 단계 → 설계 단계 → 구현 단계 → 시험 단계 → 운용/유지보수 단계

2. 정보공학 방법론 : 정보 시스템의 개발을 위해 계획, 분석, 설계, 구축에 정형화된 기법들을 상호 연관성 있게 통합 및 적용하는 자료(Data) 중심의 방법론이다
   - 정봇 시스템 개발 주기를 이용하여 대규모 정보 시스템을 구축하는데 적합
   - 데이터베이스 설계를 위한 데이터 모델링으로 개체 관계도 (ERD; Entity-Relationship Diagram)을 사용
   - 정보공학 방봅론의 절차
   : 정보 전략 계획 수립 단계 → 업무 영역 분석 단계 → 업무 시스템 설계 단계 → 업무 시스템 구축 단계

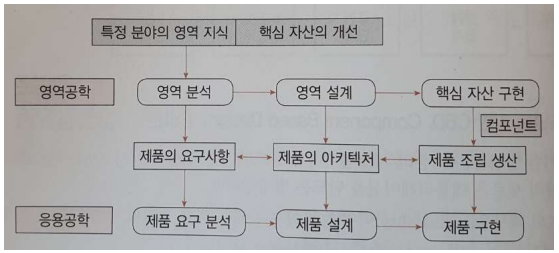
3. 객체지향 방법론 : 현실 세계의 개체를 기계의 부품처럼 하나의 객체로 만들어, 소프트웨어를 개발할 때 기계의 부품을 조립하듯이 객체들을 조립해서 필요한 소프트웨어를 구현하는 방법론
   - 객체지향 방법론은 구조적 기법의 문저점으로 인한 소프트웨어 위기의 해결책으로 채택되었다
   - 설계 과정에서 주로 사용되는 모델링 언어에는 패키지 다이어그램, 배치 다이어그램(Deployment Diagram), 상태 전이도(State Transition Diagram) 등이 있다
   - 객체지향 방법론의 절차
   : 요구 분석 단계 → 설계 단계 → 구현 단계 → 테스트 및 검증 단계 → 인도 단계

4. **컴포넌트 기반(CBD; Component Based Design) 방법론 :** 기존의 시스템이나 소프트웨어를 구성하는 컴포넌트를 조합하여 하나의 새로운 애플리케이션을 만드는 방법론
   - 컴포넌트의 재사용이 가능하여 시간과 노력을 절감할 수 있다
   - 새로운 기능을 추가하는 것이 간단하여 확장성이 보장된다
   - 유지 보수 비용을 최소화하고 생산성 및 품질을 향상시킬 수 있다.
   - 컴포넌트 기반 방법론의 절차
   : 개발 준비 단계 → 분석 단계 → 설계 단계 → 구현 단계 → 테스트 단계 → 전개 단계 → 인도 단계

5. 애자일(Agile) 방법론 : 애자일은 '민첩한', '기민한' 이라는 의미로, 애자일 방법론은 고객의 요구사항 변화에 유연하게 대응할 수 있도록 일정한 주기를 반복하면서 개발 과정을 진행하는 방법론이다
   - 소규모 프로젝트, 고도로 숙달된 개발자, 급변하는 요구사항에 적합하다
   - 애자일 방법론의 대조적인 종류에는 익스트림 프로그래밍(eXtreme Programming), 스키럼, 칸반, 크리스탈 등이 있다
   - 애자일 방법론의 절차
   : 반복 주기 — 사용자 스토리 → 계획 → 개발 → 승인 테스트

6. 제품 계열 방법론 : 특성 제품에 적용하고 싶은 공통된 기능을 정의하여 개발하는 방법론

- 임베디드 소프트웨어를 만드는데 적합하다
- 제품 계열 방법론은 영역공학과 응용공학으로 구분된다
  - ○ 영역공학 : 영역 분석, 영역 설계, 핵심 자산을 구현하는 영역
  - ○ 응용공학 : 제품 요구 분석, 제품 설계, 제품을 구현하는 영역
- 영역공학과 응용공학의 연계를 위해 제품의 요구사항, 아키텍처, 조립 생산이 필요하다
- 제품 계열 방법론의 절차



섹션 167 S/W 공학의 발전적 추세

소프트웨어 재사용의 개요 : 이미 개발되어 인정받은 소프트웨어의 전체 혹은 일부분을 다른 소프트웨어 개발이나 유지에 사용하는 것

- 소프트웨어 개발의 품질과 생산성을 높이기 위한 방법으로, 기존에 개발된 소프트웨어와 경험, 지식 등을 새로운 소프트웨어에 적용한다
- 재사용의 이점
  - ○개발 시간과 비용을 단축
  - ○소프트웨어 품질을 향상
  - ○소프트웨어 개발의 생선성을 향상
  - ○프로젝트 실패의 위험을 감소
  - ○시스템 구축 방법에 대한 지식 공유
  - ○시스템 명세, 설계, 코드 등 문서를 공유

소프트웨어 재사용 방법

합성 중심 (Composition-Based)	전자 칩과 같은 소프트웨어 부품, 즉 블록(모듈)을 만들어서 끼워 맞추어 소프트웨어를 완성시키는 방법으로, 블록 구성 방법이라고도 한다.
생성 중심 (Generation-Based)	추상화 형태로 쓰여진 명세를 구체화하여 프로그램을 만드는 방법으로, 패턴 구성 방법이라고도 한다.

소프트웨어 재공학의 개요★★★

: 새로운 요구에 맞도록 기존 시스템을 이용하여 보다 나은 시스템을 구축하고, 새로운 기능을 추가하여 소프트웨어 성능을 향상 시키는 것

- 유지보수 비용이 소프트웨어 개발 비용의 대부분을 차지하는 문제를 염두에 두어 기존 소프트웨어의 데이터와 기능들을 개조 및 개선을 통해 유지보수성과 품질을 향상 시키려는 기술이다
- 기존 소프트웨어의 기능을 개조하거나 개선하므로, 예방(Preventive) 유지보수 측면에서 소프트웨어 위기를 해결하는 방법이라고 할 수 있다
- 소프트웨어의 수명이 연장되고, 소프트웨어 기술이 향상될 뿐만 아니라 소프트웨어 개발 기간도 단축된다
- 소프트웨어에서 발생할 수 있는 오류가 줄어들고, 비용이 절감된다
- 주요 활동

분석 (Analysis)	○기존 소프트웨어의 명세서를 확인하여 소프트웨어의 동작을 이해하고, 재공학할 대상을 선정하는 활동
재구성 (Restrucuting)	○기존 소프트웨어의 구조를 향상시키기 위하여 코드를 재구성 ○소프트웨어의 기능과 외적인 동작은 바뀌지 않는다
역공학 (Reverse Engineering)	○기존 소프트웨어를 분석하여 소프트웨어 개발 과정과 데이터 처리 과정을 설명하는 분석 및 설계 정보를 재발견하거나 다시 만들어내는 활동 ○일반적인 개발 단계와는 반대 방향으로 기존 코드를 복구하거나, 기존 소프트웨어의 구성 요소와 그 관계를 파악하여 설계도를 추출
이식 (Migration)	기존 소프트웨어를 다른 운영체제나 하드웨어 환경에서 사용할 수 있도록 변환하는 활동

CASE의 개요★★★★★

: CASE(Computer Aided Software Engineering)는 소프트웨어 개발 과정에서 사용되는 요구 분석, 설계, 구현, 검사 및 디버깅 과정 전체 또는 일부를 컴퓨터와 전용 소프트웨어 도구를 사용하여 자동화하는 것이다.

- 객체지향 시스템, 구조적 시스템 등 다양한 시스템에서 활용되는 자동화 도구(CASE Tool)이다
- CASE 도구를 통해 관리되는 공통 모듈을 사용할 수 있어 재사용성 향상
- CASE 도구가 모듈 관리를 자동으로 수행하므로 유지보수가 간편
- 소프트웨어 개발 도구와 방법론이 결합된 것으로, 정형화된 구조 및 방법(매커니즘)을 소프트웨어 개발에 적용하여 생산성 및 품질 향상을 구현느 공학 기법
- 소프트웨어 개발의 모든 단계에 걸쳐 일관된 방법론을 제공하는 자동화 도구들을 지원하고, 개발자들은 이 도구를 사용하여 소프트웨어 개발의 표준화를 지향하며, 자동화의 이점을 얻을 수 있게 해준다

- CASE 도구는 요구 분석, 설계 과정을 지원하는 상위 CASE도구와 구현, 테스트 과정을 지워나는 하위 CASE 도구로 구분할 수 있다.
- **CASE 주요 기능** : 소프트웨어 생명 주기 전 단계의 연결, 다양한 소프트웨어 개발 모형 지원, 그래픽 지원, 모델들의 모순 검사 및 오류검증, 자료흐름도 작성 등
- **CASE의 원천 기술** : 구조적 기법 프로토타이핑, 자동 프로그래밍, 정보 저장소, 분산처리

섹션 168 비용 산정 기법(C)

: 소프트웨어 비용 산정은 소프트웨어의 개발 규모를 소요되는 인원, 자원, 기간 등으로 확인하여 실행 가능한 계획을 수립하기 위해 필요한 비용을 산정하는 것
- 소프트웨어 비용 산정을 너무 높게 산정할 경우 예산 낭비와 일의 효율성 저하를 초리해라 수 있고, 너무 낮게 산정할 경우 개발자의 부담이 가중되고 품질문제가 발생할 수 있다
- 소프트웨어 비용 산정 기법에는 하향식 비용 산정 기법과 상향식 비용 산정 기법이 있음

소프트웨어 비용 결정 요소
1. 프로젝트 요소
- 제품 복잡도 : 소프트웨어의 종류에 따라 발생할 수 있는 문제점들의 난이도
- 시스템 크기 : 소프트웨어의 규모에 따라 개발해야할 시스템의 크기를 의미
- 요구되는 신뢰도 : 일정 기간 내 주어진 조건하에서 프로그램이 필요한 기능을 수행하는 정도를 의미

2. 자원요소
- 인적 자원 : 소프트웨어 개발자들이 갖춘 능력 혹은 자질을 의미
- 하드웨어 자원 : 소프트웨어 개발 시 필요한 장비와 워드프로세서, 프린터 등의 보조 장비를 의미

- 소프트웨어 자원 : 소프트웨어 개발 시 필요한 언어 분석기, 문서화 도구 등의 개발 지원 도구를 의미
3. 생산성 요소
- 개발자 능력 : 개발자들이 갖춘 전문지식, 경험, 이해도, 책임감 , 창의력 등을 의미
- 개발 기간 : 소프트웨어를 개발하는 기간을 의미

섹션 168 비용 산정 기법 - 하향식 (C)

: 하향식 비용 산정 기법은 과거의 유사한 경험을 바탕으로 전문 지식이 많은 개발자들이 참여한 회의를 통해 비용을 산정하는 비과학적인 방법이다.
- 프로젝트의 전체 비용을 산정한 후 각 작업별로 비용을 세분화한다
- 하향식 비용 산정 기법에는 전문가 감정 기법, 델파이 기법 등이 있다

1. 전문가 감정 기법 : 조직 내에 있는 경험이 많은 두 명 이상의 전문가에게 비용 산정을 의뢰하는 기법이다
- 가장 편리하고 신속하게 비용을 산정할 수 있으며, 의뢰자로부터 믿음을 얻을 수 있음
- 새로운 프로젝트에는 과거의 프로젝트와 다른 요소들이 있다는 것을 간과할 수 있음
- 새로운 프로젝트와 유사한 프로젝트에 대한 경험이 벗을 수 있음
- 개인적이고 주관적일 수 있다

2. 델파이 기법 : 전문가 감정 기법의 주관적인 편견을 보완하기 위해 많은 전문가의 의견을 종합하여 산정하는 기법이다
- 전문가들의 편견이나 분위기에 지배되지 않도록 한 명의 조정자와 여러 전문가로 구성된다
- 비용 산정 순서
  ①조정자는 각 비용 산정 요원에게 시스템 의서와 산정한 비용 내역을 기록할 서식을 제공
  ②산정 요원들은 정의서를 분석하여 익명으로 그들 나름대로의 비용 산정
  ③조정자는 산정 요원들의 반응을 요약하여 배포
  ④산정 요원들은 이전에 산정한 결과를 이용하여 다시 익명으로 산정
  ⑤요원들 간의 의견이 건의 일치할 때까지 이 과정을 반복

섹션 170 비용 산정 기법 - 상향식 (A)

: 프로젝트의 세부적인 작업 단위별로 비용을 산정한 후 집계하여 전체 비요을 산정하는 기법이다

- LOC(원시 코드 라인 수) 기법, 개발 단계별 인월수 기법, 수학적 산정 기법등이 있음

1. LOC(원시 코드 라인 수, source Line Of Code) 기법

: 소프트웨어의 각 기능의 원시 코드 라인 수의 비관치, 낙관치, 기대치를 측정하여 예측치를 구하고 이를 이용하여 비용을 산정하는 기법

- 측정이 용이하고 이해하기 쉬워 가장 많이 사용된다
- 예측치를 이용하여 생산성, 노력 , 개발 기간 등의 비용을 산정

$$\text{예측치} = \frac{a + 4m + b}{6}$$ 단, a: 낙관치, b: 비관치 , m : 기대치(중간치)

- 산정 공식

○**노력(인월)** = 개발 기간 X 투입 인원 = LOC/1인당 월평균 생산 코드 라인 수
○**개발 비용 =** 노력(인월) X 단위 비용(1인당 월평균 인건비)
○**개발 기간 =** 노력(인월) / 투입 인원
○**생산성** = LOC / 노력(인월)

2. 개발 단계별 인월수 (Effort Per Task) 기법

: LOC 기법을 보완하기 위한 기법으로, 각 기능을 구현시키는데 필요한 노력을 생명 주기의 각 단계별로 산정한다

- LOC 기법보다 좀 더 정확함

섹션 171 수학적 산정 기법

: 수학적 산정 기법은 상향식 비용 산정 기법으로, 경험적 추정 모형, 실험적 추정 모형이라고도 하며 개발 비용 산정의 자동화를 목표로한다

- 비용을 자동적으로 산정하기 위해 사용되는 공식은 과거 유사한 프로젝트를 기반으로항여 경험적으로 유도된 것이다.
- 수학적 산정 기법에는 COCOMO 모형, Putnam 모형, 기능 점수(FP) 모형 등이 있으며 각 모형에서는 지정된 공식을 이용해 비용을 산정한다

COCOMO 모형의 개요 : COCOMO(COnstructive COst MOdel) 모형은 보헴(Boehm)이 제안한 것으로, 우너시 프로그램의 규모인 LOC에 의한 비용 산정 기법이다.

- 개발할 소프트웨어의 규모(LOC)를 예측한 후 이를 소프트웨어 종류에 따라 다르게 책정되는 비용 산정 방정식에 대입하여 비용을 산정한다
- 비교적 작은 규모의 프로젝트들을 통계 분석한 결과를 반영한 모델이므로 중소규모 소프트웨어 프로젝트 비용 추정에 적합하다
- 같은 규모의 프로그램이라도 그 성격에 따라 비용이 다르게 산정된다
- 비용 산정 결과는 프로젝트를 완성하는데 필요한 노력(Man-Month)로 나타낸다

COCOMO의 소프트웨어 개발 유형

: 소프트웨어 개발 유형은 소프트웨어의 복잡도 혹은 원시 프로그램의 규모에 따라 조직형, 반분리형, 내장형으로 분류할 수 있다.

1. **조직형 (Organic Mode) :** 기관 내부에서 개발된 중·소 규모의 소프트웨어로 일괄 자료 처리나 과학 기술 계산용, 비즈니스 자료 처리용으로 5만(50KDSI) 라인 이하의 소프트웨어를 개발하는 유형이다.

- 사무 처리용, 업무용, 과학용 응용 소프트웨어 개발에 적합하다
- 비용을 산정하는 공식은 다음과 같다.

○노력(MM) = 2.4 X $(KDSI)^{1.05}$
○개발 기간(TDEV) = 2.5 X $(MM)^{0.38}$

2. **반분리형 (Semi-Detached Mode)** : 조직형과 내장형의 중간으로 트랜잭션 처리 시스템이나 운영체제, 데이터베이스 관리 시스템 등의 30만(300KDSI) 라인 이하의 소프트웨어를 개발하는 유형이다

- 컴파일러, 인터프리터와 같은 유틸리티 개발에 적합하다
- 비용을 산정하는 공식은 다음과 같다

○노력(MM) = 3.0 X $(KDSI)^{1.12}$
○개발 기간(TDEV) = 2.5 X $(MM)^{0.35}$

3. **내장형 (Embedded Mode)** : 초대형 규모의 트랜잭션 처리 시스템이나 운영체제 등의 30만(300KDSI) 이상의 소프트웨어를 개발하는 유형

- 신호기 제어 시스템, 미사일 유도 시스템, 실시간 처리 시스템 등의 시스템 프로그램 개발에 적합하다
- 비용을 산정하는 공식은 다음과 같다
  ○노력(MM) = 3.6 X $(KDSI)^{1.20}$
  ○개발 기간(TDEV) = 2.5 X $(MM)^{0.32}$

PutNam 모형 : 소프트웨어 생명 주기의 전 과정 동안에 사용될 노력의 분포를 가정해주는 모형이다
- 푸트남(Putnam)이 제안한 것으로 생명 주기 예측 모형이라고도 한다
- 시간에 따른 함수로 표현되는 Rayleigh-Norden 곡선의 노력 분포도를 기초로 한다
- 대형 프로젝트의 노력 분포 산정에 이용되는 기법이다
- 개발 기간이 늘어날수록 프로젝트 적용 인원의 노력이 감소한다
- 산정 공식

$$\text{개발 노력(MM)} = \frac{L^3}{C_k^3 \cdot Td^4}$$

- L : 원시 코드 라인 수
- Td : 개발 기간
- C_k : 환경 상수(빈약 환경 = 2,000, 중간 환경 = 8,000, 최적 환경 = 12,000)

기능 점수 모형 (FP) : 알브레히트(Albrecht)가 제안한 것으로, 소프트웨어의 기능을 증대시키는 요인별로 가중치를 부여하고, 요인별로 가중치를 부여하고ㅓ, 요인별 가중치를 합산하여 총 기능 점수를 산출하여 총 기능 점수와 영향도를 이용하여 기능 점수(FP)를 구한 후 이를 이용해서 비용을 산정하는 기법이다

기능 점수(FP) = 총 기능 점수 X [0.65 + (0.1 X 총 영향도)]

- 발표 초기에는 관심을 받지 못하였으나 최근에는 그 유용성과 간편성으로 비용 산정 기법 가운데 최선의 평가를 받고 있다
- **기능별 가중치**

소프트웨어 기능 증대 요인	가중치		
	단순	보통	복잡
자료 입력 (입력 양식)	3	4	6
정보 출력 (출력 보고서)	4	5	7
명령어 (사용자 질의 수)	3	4	5
데이터 파일	7	10	15
필요한 외부 루틴과의 인터페이스	5	7	10

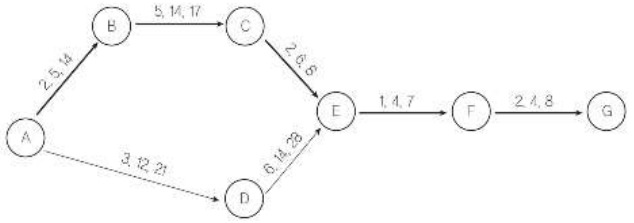
※자동화 추정 도구
: 비용 산정의 자동화를 위해 개발된 도구로는 SLIM과 ESTIMACS가 있다
- SLIM : Rayleigh-Norden 곡선과 Putnam 예측 모델을 기반으로 개발된 자동화 추정 도구
- ESTIMACS : 따양한 프로젝트와 개인별 요소를 수용하도록 FP 모형을 기초로 하여 개발된 자동화 추정 도구

섹션 172 프로젝트 일정 계획

: 프로젝트의 프로세스를 이루는 소작업을 파악하고 예측된 노력을 각 소작업에 분배하여, 소작업의 순서와 일정을 정하는 것이다
- 소프트웨어 개발 기간의 지연을 방지하고 프로젝트가 계획대로 진행되도록 일정을 계획한다
- 계획된 일정은 프로젝트의 진행을 관리하는데 기초 자료가 된다
- 계획된 일정과 프로젝트의 진행도를 비교하여 차질이 있을 경우 여러 조치를 통해 조정할 수 있다
- 프로젝트 일정 계획을 위해서는 WBS, PERT/CPM, 간트 차트 등이 사용된다

1. PERT(Program Evaluatiuon and Review Technique, 프로그램 평가 및 검토 기술) : 프로젝트에 필요한 전체 작업의 상호 관계를 표시하는 네트워크로 각 작업별로 낙관적인 경우, 가능성이 있는 경우, 비관적인 경우로 나누어 각 단계별 종료 시기를 결정하는 방법이다
- 과거에 경험이 없어서 소요 기간 예측이 어려운 소프트웨어에서 사용
- 노드와 간선으로 구성되며 원 노드에는 작업을, 간선(화살표)에는 낙관치, 기대치, 비관치 등을 표시한다
- 결정 경로, 작업에 대한 경계 시간, 작업 간의 상호 관련성 등을 알 수 있다.
- 다음과 같은 공식으로 작업 예측치를 계산한다

$$\text{작업예측치} = \frac{\text{비관치} + 4*\text{기대치} + \text{낙관치}}{6}, \quad \text{평방편차} = [\frac{(\text{비관치} - \text{낙관치})}{6}]^2$$

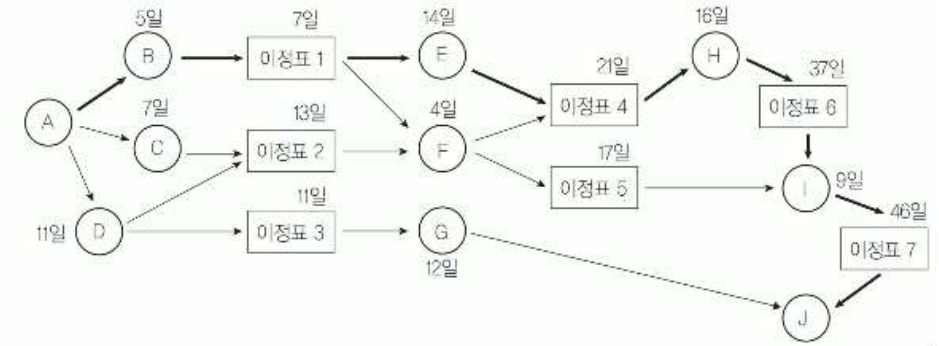


2. CPM(Critical Path Method, 임계 경로 기법)

: 프로젝트 완성에 필요한 작업을 나열하고 작업에 필요한 소요 기간을 예측하는데 사용하는 기법이다

- CPM은 노드와 간선으로 구성된 네트워크로 노드는 작업을, 간선은 작업 사이의 전후 의존 관계를 나타낸다
- 원형 노드는 각 작업을 의미하며 각 작업 이름과 소요 기간을 표시하고, 박스 노드는 이정표를 의미하며 박스 노드 위에는 예상 완료 시간을 표시
- 간선을 나타내는 화살표의 흐름에 따라 각 작업이 진행되며, 전 작업이 완료된 후 다음 작업을 진행할 수 있음
- 각 작업의 순서와 의존 관계, 어느 작업이 동시에 수행될 수 있는지를 한눈에 볼 수 있음
- 경영층의 과학적인 의사 결정을 지원하며, 효과적인 프로젝트의 통제를 가능하게 함
- 병행작업이 가능하도록 계획 가능하며, 이를 위한 자원 할당도 가능
- 임계 경로는 최장 경로를 의미

- 그 후 결정된 일정 계획을 간트 차트로 나타냄



3. **간트 차트(Gantt Chart)**

: 프로젝트의 각 작업들이 언제 시작하고 언제 종료되는지에 대한 작업 일정을 막대 도표를 이용하여 표시하는 프로젝트 일정표로, 시간선(Time-Line) 차트라고도 한다

- 중간 목표 미달성 시 그 이유와 기간을 예측할 수 있게 한다
- 사용자와의 문제점이나 예산의 초과 지출 등도 관리할 수 있다.
- 자원 배치와 인원 계획에 유용하게 사용된다
- CPM 네트워크의 데이터를 기반으로 간트 차트를 제작 가능
- 작업 경로는 표시할 수 없으며, 계획의 변화에 대한 적응성이 약하다
- 계획 수립 또는 수정 때 주관적인 수치에 기울어지기 쉽다
- 간트 차트는 이정표, 작업 일정, 작업 기간, 산출물로 구성되어 있다
- 수평 막대의 길이는 각 작업의 기간을 나타낸다

섹션 173 소프트웨어 개발 방법론 결정

: 프로젝트 관리와 재사용 현황을 소프트웨어 개발방법론에 반영하고, 확정된 소프트웨어 생명 주기와 개발 방법론에 맞춰 소프트웨어 개발 단계, 활동, 작업, 절차등을 정의함

※프로젝트 관리 : 주어진 기간 내에 최소의 비용으로 사용자를 만족시키니는 시스템을 개발하기 위한 전반적인 활동

관리 유형	주요 내용
일정 관리	작업 순서, 작업 기간 산정, 일정 개발, 일정 통제
비용 관리	비용 산정 , 비용 예산 편성, 비용 통제
인력 관리	프로젝트 팀 편성, 자원 산정, 프로젝트 조직 정의, 프로젝트 팀 개발, 자원 통제, 프로젝트 팀 관리
위험 관리	위험 식별, 위험 평가, 위험 대처, 위험 통제
품질 관리	품질 계획, 품질 보증 수행, 품질 통제 수행

소프트웨어 개발 방법론 결정 절차

1. 프로젝트 관리와 재사용 현황을 소프트웨어 개발 방법론에 반영
2. 개발 단계별 작업 및 절차를 소프트웨어 생명 주기에 맞춰 수립
3. 결정된 소프트웨어 개발 방법론의 개발 단계별 활동 목적, 작업 내용, 산출물에 대한 매뉴얼을 작성

섹션 174 소프트웨어 개발 표준

: 개발 단계에서 수행하는 품질 관리에 사용되는 국제 표준을 의미

- 대표적인 소프트웨어 개발 표준에는 ISO/IEC 12207, CMMI, SPICE 등

ISO/IEC 12207 : ISO에서 만든 표준 소프트웨어 생명 주기 프로세스로, 소프트웨어의 개발, 운영, 유지보수 등을 체계적으로 관리하기 위한 소프트웨어 생명 주기 표준을 제공한다

- ISO/IEC 12207은 기본 생명 주기 프로세스, 자원 생명 주기 프로세스, 조직

생명 주기 프로세스로 구분한다.

기본 생명 주기 프로세스	획득, 공급, 개발, 운영, 유지보수 프로세스
자원 생명 주기 프로세스	품질 보증, 검증, 확인, 활동 검토, 감사, 문서화, 형상 관리, 문제 해결 프로세스
조직 생명 주기 프로세스	관리, 기반 구조, 훈련 , 개선 프로세스

CMMI(Capability Maturity Model Integration)

: CMMI(능력 성숙도 통합 모델)은 소프트웨어 개발 조직의 업무 능력 및 조직의 성숙도를 평가하는 모델로, 미국 카네기멜론 대학교의 소프트웨어 공학연구소(SEI)에서 개발하였음

- CMMI의 소프트웨어 프로세스 성숙도는 초기, 관리, 정의, 정량적 관리, 최적화 5단계로 구분한다

단계	프로세스	특징
초기	정의된 프로세스 없음	작업자 능력에 따라 성공 여부 결정
관리	규칙화된 프로세스	특정한 프로젝트 내의 프로세스 정의 및 수행
정의	표준화된 프로세스	조직의 표준 프로세스를 활용하여 업무 수행
정량적 관리	예측 가능한 프로세스	프로젝트를 정량적으로 관리 및 통제
최적화	지속적 개선 프로세스	프로세스 역량 향상을 위한 지속적인 프로세스 개선

SPICE (Software Process Improvement and Capability dEtermination)

: SPICE(소프트웨어 처리 개선 및 능력 평가 기준)은 정보 시스템 분야에서 소프트웨어의 품질 및 생산성 향상을 위해 소프트웨어 프로세스를 평가 및 개선하는 국제 표준으로 , 공식 명칭 ISO/IEC 15504 이다

SPICE의 목적

- 프로세스 개선을 위해 개발 기관이 스스로 평가하는 것
- 기관에서 지정한 요구조건의 만족 여부를 개발 조직이 스스로 평가하는 것
- 계약 체결을 위해 수탁 기관의 프로세스를 평가하는 것

SPICE는 5개의 프로세스 범주와 40개의 세부 프로세스로 구성됨

범주	특징
고객-공급자 프로세스	○소프트웨어를 개발하여 고객에게 전달하는 것을 지원하고, 소프트웨어의 정확한 운용 및 사용을 위한 프로세스로 구성됨 ○구성 요소 : 인수, 공급, 요구, 도출, 운영 ○프로세스 수 : 10개
공학 프로세스	○시스템과 소프트웨어 제품의 명세화, 구현, 유지보수를 하는데 사용하는 프로세스로 구성됨 ○구성 요소 : 개발, 소프트웨어 유지보수 ○프로세스 수 : 9개
지원 프로세스	○소프트웨어 생명 주기에서 다른 프로세스에 의해 이용되는 프로세스들로 구성됨 ○구성 요소 : 문서화,형상, 품질 보증, 검증. 확인, 리뷰, 감사, 품질 문제 해결 ○프로세스 수 : 8개
관리 프로세스	○소프트웨어 생명 주기에서 프로젝트 관리에 의해 사용되는 프로세스로 구성됨 ○구성 요소 : 관리, 프로젝트 관리, 품질 및 위험 관리 ○프로세스 수 : 4개
조직 프로세스	○조직의 업무 목적 수립과 조직의 업무 목표 달성을 위한 프로세스로 구성됨 ○구성 요소 : 조직 배치, 개선 활동 프로세스, 인력 관리, 기반 관리, 측정 도구, 재사용 ○프로세스 수 : 9개

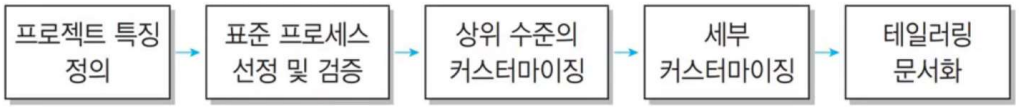
SPICE는 프로세스 수행 능력 단계를 6단계로 나눈다

단계	특징
Level 0 - 불완전 (Imcomplete)	프로세스가 구현되지 않았거나 목적을 달성하지 못한 단계
Level 1 - 수행 (Performed)	프로세스 수행되고 목적이 달성된 단계
Level 2 - 관리 (Managed)	정의된 자원 한도 내에서 그 프로세스가 작업 산출물로 인도하는 단계
Level 3 - 확립 (Established)	소프트웨어 공학 원칙에 기반하여 정의된 프로세스가 수행되는 단계
Level 4 - 예측 (Predictable)	프로세스가 목적 달성을 위해 통제되고, 양적인 측정을 통해서 일관되게 수행되는 단계
Level 5 - 최적화 (Optimizing)	프로세스 수행을 최적화하고, 지속적인 개선을 통해 업무 목적을 만족시키는 단계

섹션 175 **소프트웨어 개발 방법론 테일러링**

: 프로젝트의 상황 및 특성에 맞도록 정의된 소프트웨어 개발 방법론의 절차, 사용 기법 등을 수정 및 보완하는 작업

- 관리적 측면에서 볼 때 테일러링은 최단기간에 안정적인 프로젝트 진행을 위해 사전 위험을 식별하고 제거하는 작업이다
- 기술적 측면에서 볼 때 테일러링은 프로젝트에 최적화된 기술 요소를 도입하여 프로젝트 특성에 맞는 최적의 기법과 도구를 찾아가는 과정이다
- 소프트웨어 개발 방법론 테일러링 수행 절차



소프트웨어 개발 방법론 테일러링 고려사항

- **내부적 기준**
  - ○목표 환경 : 시스템의 개발 환경과 유형이 서로 다른 경우 테일러링이 필요
  - ○요구사항 : 프로젝트의 생명 주기 활동에서 개발, 운영, 유지보수 등 프로젝트에서 우선적으로 고려할 요구사항이 서로 다른 경우 테일러링이 필요
  - ○프로젝트 규모 : 비용, 인력, 기간 등 프로젝트의 규모가 서로 다른 경우
  - ○보유 기술 : 프로세스, 개발 방법론, 산출물, 구성원의 능력 등이 서로 다른 경우 테일러링이 필요
- **외부적 기준**
  - ○법적 제약사항 : 프로젝트별로 적용될 IT Compliance가 서로 다른 경우
  - ○표준 품질 기준 : 금융, 제도 등 분야별 표준 품질 기준이 서로 다른 경우

소프트웨어 개발 방법론 테일러링 기법

- 프로젝트 규모와 복잡도에 따른 테일러링 기법
- 프로젝트 구성원에 따른 테일러링 기법
- 팀내 방법론 지원에 따른 테일러링 기법
- 자동화에 따른 테일러링 기법

섹션 176 소프트웨어 개발 프레임워크 ★★★

: **프레임워크**는 소프트웨어 개발에 공통적으로 사용되는 구성 요소와 아키텍처를 일반화하여 손쉽게 구현할 수 있도록 여러 가지 기능들을 제공해주는 반제품 형태의 소프트웨어 시스템이다

- 선행 사업자의 기술에 의존하지 않은 표준화된 개발 기반으로 인해 사업자 종속성이 해소된다
- 개발해야 할 애플리케이션의 일부분이 이미 내장된 클래스 라이브러리로 구현되어 있어 개발자는 이미 존재하는 부분을 확장 및 이용하는 것으로 소프트웨어를 개발할 수 있다
- 프레임워크의 주요 기능에는 예외 처리, 트랜잭션 처리, 메모리 공유, 데이터 소스 관리, 서비스 관리, 쿼리 서비스, 로깅 서비스, 사용자 인증 서비스 등이 있다
- 프레임워크의 종류에는 스프링 프레임워크, 전자정부 프레임워크, 닷넷 프레임워크 등이 있다.

※프레임워크의 특성

모듈화 (Modularity)	○프레임워크는 캡슐화를 통해 모듈화를 강화하고 설계 및 구현의 변경에 따른 영향을 최소화함으로써 소프트웨어의 품질을 향상시킴 ○프레임워크는 개발표준에 의한 모듈화로 인해 유지보수가 용이
재사용성 (Reusability)	○프레임워크는 재사용 가능한 모듈들을 제공함으로써 예산 절감, 생산성 향상, 품질 보증이 가능
확장성 (Extensibility)	○프레임워크는 다형성을 통한 인터페이스 확장이 가능하여 다양한 형태와 기능을 가진 애플리케이션 생산 가능
제어의 역흐름 (Inversion of Control)	○개발자가 관리하고 통제해야 하는 객체들의 제어를 프레임워크에 넘김으로써 생산성을 향상

1. 스프링 프레임워크
   - 자바 플랫폼을 위한 오픈 소스 경량형 애플리케이션 프레임워크
   - 동적인 웹 사이트의 개발을 위해 다양한 서비스 제공
   - 전자정부 표준 프레임워크의 기반 기술로 사용

2. 전자 정부 프레임워크
   - 우리나라의 공공부문 정보화 사업 시 효율적인 정보 시스템의 구축을 지원하기 위해 필요한 기능 및 아키텍처를 제공하는 프레임워크
   - 전자정부 프레임워크는 개발 프레임워크의 표준 정립으로 응용 소프트웨어의 표준화, 품질 및 재사용성의 향상을 목적으로 함

- 전자정부 프레임워크는 오픈 소스 기반의 범용화가 되고 공개된 기술을 활용함으로써 특정 업체의 종속성을 배제하고 사업별 공통 컴포넌트의 중복 개발을 방지

3. 닷넷 프레임워크(.NET Framework)
- Windows 프로그램의 개발 및 실행 환경을 제공하는 프레임워크로, Microsoft 사에서 통합 인터넷 전략을 위해 개발
- 닷넷 프레임워크는 코드 실행을 관리하는 CLR(Common Language Runtime, 공용 언어 런타임)이라는 이름의 가상머신 상에서 작동
- 닷넷 프레임워크는 메모리 관리, 유형 및 메모리 안정성, 보안, 네트워크 작업 등 여러 가지 서비스 제공

2장 IT프로젝트 정보시스템 구축 관리

섹션 177 네트워크 관련 신기술

지능형 초연결망

: 과학기술정보통신부 주관으로 추진중인 사업으로, 스마트 시티, 스마트 스테이션 등 4차 산업혁명 시대를 자아 새로운 변화에 따라 급격하게 증가하는 데이터 트래픽을 효과적으로 수용하기 위해 시행되는 정부 주관 사업이다

- 지능형 초연결망은 국가 전체 망에 소프트웨어 정의 기술(SDE)을 적용하는 방법으로 네트워크의 데이터 트래픽 증가를 불러올 수 있는 사물 인터넷(IoT), 클라우드, 빅데이터, 5G 등을 효율적으로 수용할 수 있도록 한다
- 지능형 초연결망은 기존의 초고속정보통신망, 광대역통합망(BcN), 광대역융합망(UBcN)을 잇는 중장기 네트워크 발전 전략이다

소프트웨어 정의 기술(SDE,SDX; Software-Defined Everything)

: 소프트웨어 정의 기술은 네트워크, 데이터 센터 등에서 소유한 자원을 가상화하여 개별 사용자에게 제공하고, 중앙에서는 통합적으로 제어가 가능한 기술이다

- 관련 용어

소프트웨어 정의 네트워킹 (SDN; Software Defined Networking)	○네트워크를 컴퓨터처럼 모델링하여 여러 사용자가 각각의 소프트웨어들로 네트워킹을 가상화하여 제어하고 관리하는 네트워크이다 ○하드웨어에 의;존하는 네트워크 체계에 비해 보다 효율적으로 네트워크를 제어,관리할 수 있다 ○기존 네트워크에는 영향을 주지 않으면서 특정 서비스의 전송 경로 수정을 통하여 인터넷상에서 발생하는 문제를 해결할 수 있음
소프트웨어 정의 데이터 센터 (SDDC; Software Dfined Data Center)	데이터 센터의 모든 자원을 가상화하여 인력의 개입없이 소프트웨어 조작만으로 관리 및 제어ㅓ되는 데이터 센터이다
소프트웨어 정의 스토리지 (Software Defined Storage)	물리적인 데이터 스토리지를 가상화하여 여러 스토리지를 하나처럼 관리하거나, 하나의 스토리지를 여러 스토리지로 나눠 사용할 수 있는 기술

IoT(Internet of Things , 사물 인터넷)

: IoT는 정보 통신 기술을 기반으로 실세께(Physical World)와 가상 세계 (Virtual World)의 다양한 사물들을 인터넷으로 서로 연결하여 진보된 서비스를 제공하기 위한 서비스 기반 기술이다.

- 유비쿼터스 공간을 구현하기 위한 컴퓨팅 기기들이 환경과 사물에 심겨 환경이나 사물 그 자체가 지능화되는 것부터 사람과 사물, 사물과 사물 간에 지능 통신을 할 수 있는 엠투엠(M2M; Machine to Machine)의 개념을 인터넷으로 확장하여 사물은 물론, 현실과 가상 세계의 모든 정보와 상호 작용하는 IoT 개념으로 진화함
- IoT의 주요 기술로는 스마트 샌싱 기술, 유무선 통신 및 네트워크 인프라 기술, 사물 인터넷 인터페이스 기술, 사물 인터넷을 통한 서비스 기술 등이 있다
- IoT 기반 서비스는 개방형 아키텍처를 필요로하기 때문에 정보 공유에 대한 부작용을 최소화하기 위한 정보 보안 기술의 적용이 중요

IoT 관련 용어

용어	의미
M2M (사물 통신)	○무선 통신을 이용한 기계와 기계 사이의 통신 ○변압기 원격 감시, 전기, 가스 등의 원격 검침, 무선 신용카드 조회기, 무선 보안 단말기, 버스 운행 시스템, 위치 추적 시스템, 시설물 관리 등을 무선으로 통합하여 상호 작용한다.
메시 네트워크 (Mesh Network)	○차세대 이동통신, 홈네트워킹, 공공 안전 등 특수 목적을 위한 네트워크 ○수십에서 수천 개의 디바이스를 그물망과 같이 유기적으로 연결하여 모든 구간을 동일한 무선망처럼 구성하여 사용자가 안정적인 네트워크를 사용할 수 있도록함
와이선 (WI-SUN)	○스마트 그리드와 같은 장거리 무선 통신을 필요로 하는 사물 인터넷(IoT) 서비스를 위한 저전력 장거리(LPWA;Low-Power Wide

	Area) 통신 기술이다 ○낮은 지연 속도, 메시 네트워크 기반 확장성, 펌웨어 업그레이드 용이성 등으로 짧은 시간 동안 데이터 전송이 빈번한 검침 분야에 유용하다
UWB(Ultra WideBand, 초광대역)	○짧은 거리에서 많은 양의 디지털 데이터를 낮은 전력으로 전송하기 위한 무선 기술로 무선 디지털 퍼르ㅅ라고도 하며, 블루투스와 비교되는 기술 ○땅 속이나 벽면 뒤로도 전송이 가능하여 지진 등 재해가 일어났을 떄 전파 탐지기 기능으로 인명 구조를 할 수 있는 등 으용 범위가 광범위
피코넷 (PICONET)	○여러 개의 독립된 통신장치가 블루투스 기술이나 UBW 기술을 사용하여 통신망을 형성하는 무선 네트워크 기술 ○주로 수십 미터 이내의 좁은 공간에서 네트워크를 형성한다는 점, 정지 또는 이동 중에 있는 장치 모두를 포함한다는 특징이있음
USN (Ubiquitous Sensor Network)	○각종 센서로 다양한 정보를 무선으로 수집할 수 있도록 구성한 네트워크 ○필요한 모든 것(곳)에 RFID 태그를 부착하고, 이를 통하여 사물의 인식 정보는 물론 주변의 환경정보까지 탐지한 모든 데이터 관리 가능
SON (Self Organizing Network, 자동 구성 네트워크)	○주변 상황에 맞추어 스스로 망을 구성하는 네트워크 ○갑작스러운 사용자의 증가나 감소 시에는 자동을 주변 셀과의 협력을 통해 셀 용량을 변화시키며, 장애가 발생시 자체적인 치유도 가능
저전력 블루투스 기술 (BLE; Bluetooth Low Energy)	○일반 블루투스와 동일한 2.4GHZ 주파수 대역을 사용하지만 연결되지 않은 대기 상태에서는 절전 모드를 이용하는 방법 ○주로 낮은 전력으로 저용량 데이터를 처리하는 시계, 장난감, 비컨, 착용 컴퓨터 등의 극소형 사물 인터넷에 매우 적함
NFC (Near Field Communication) 근거리 무선 통신	○고주파(HF)를 이용한 근거리 무선 통신 기술로, 아주 가까운 거리에서 양방향 통신을 지원하는 RFID 기술의 일종 ○13.56MHZ 주파수를 이용해 10cm 내에서 최고 424Kbps의 속도로 데이터 전송을 지원하고, 모바일 기기를 통한 결제뿐만 아니라 여행 정보 전송, 교통, 출입 통제, 잠금장치 따위에 광범위하게 활용

클라우드 컴퓨팅(Cloud Computing)