1강. 소프트웨어 공학 개요

■ 학습개요

소프트웨어의 정의와 중요성을 알아보고 성질, 분류 방법 및 응용 분야를 살펴본다. 소프트웨어 공학의 탄생 배경이 된 소프트웨어 위기 현상이 무엇이었는지 확인하고 소프트웨어 공학의 명확한 정의를 숙지한다. 소프트웨어 공학 환경의 구성 요소인 원리, 방법과 기술, 방법론, 그리고 도구에 관해 명확히 구분하여 정리한다. 끝으로 좋은 소프트웨어가 갖춰야 할 여러 품질 기준들을 학습한다.

■ 학습목표

1	소프트웨어의 성질, 분류, 응용 분야를 학습한다.
2	소프트웨어 공학의 정의를 기술할 수 있다.
3	소프트웨어 공학 원리, 방법, 방법론 및 도구 간의 관계를 구별할 수 있다.
4	소프트웨어의 품질을 평가하기 위한 다양한 기준을 열거할 수 있다.

■ 주요용어

용어	해설
소프트웨어	포괄적 의미에서 프로그램과 데이터 및 관련 문서들의 묶음
소프트웨어 공학	소프트웨어 생산과 관련된 모든 부분에 공학적 원리와 방법을 적용시키려는 연구 분야 또는 전문 작업 분야
소프트웨어 위기 현상	하드웨어의 성능은 좋아지고 비용이 떨어지는데 비해 소프트웨어는 복잡해지면서 비용이 증가하는 문제를 일컫는 것
소프트웨어 개발 방법	소프트웨어 개발을 위한 구조적 방법을 말하는 것으로 따라야 할 행위들, 결과의 표기법 및 규칙 등을 포함
소프트웨어 개발 방법론	프로세스(what)와 방법(how)을 하나로 묶은 것으로 하나의 프로세스를 수행할 때 적용되어야 하는 여러 방법들과 기술들을 표현한 것
소프트웨어 개발 프로세스	소프트웨어 개발이나 유지보수 목적으로 수행되는 일련의 활동

메뉴	학습하기 (PPT 내용)

1.1 소프트웨어

소프트웨어의 정의
- 프로그램과 관련 데이터의 묶음
- 포괄적 의미의 소프트웨어는 관련 문서들을 포함한 개념

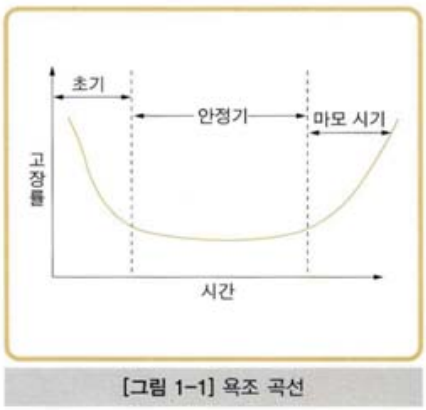
소프트웨어의 중요성
소프트웨어의 역할
- 사업체의 의사결정을 지원
- 과학적/공학적 문제 해결을 지원
- 모든 종류의 컴퓨터 시스템에 내장됨
-> 중요성의 증가, 세상을 바꾸는 동력

소프트웨어의 분류
- 시스템 소프트웨어: 컴퓨터를 동작시키기 위한 목적의 소프트웨어
- 응용 소프트웨어: 사용자의 실제 업무를 수행하는 소프트웨어

고객에 따른 소프트웨어의 분류
- 일반(generic) 소프트웨어
    - 불특정 다수를 대상으로 설계되는 소프트웨어
    - 요구사항이 일반적이고 안정적
    - 상용 소프트웨어
    - 워드 프로세서, 데이터베이스 관련 제품, 사무용 패키지, 운영체체 등
- 맞춤형(customized) 소프트웨어
    - 특정 고객을 위해 개발됨
    - 응용 도메인, 사용 환경, 요구사항이 특별함
    - 프로세스 제어, 교통 관제 시스템, 병원 관리 시스템 등

소프트웨어의 성질
- 무형의 인공물로 물질적인 성질이 없음
- 컴포넌트들의 조립을 통해 만들기 어려움
- 상대적으로 설계 과정의 품질 보증 활동이 중요함
- 개발 비용의 대부분은 노동력에 투입됨
- 상대적으로 변경이 용이함
    - 소프트웨어의 유연성 또는 순응성이라 함
- 소프트웨어는 마모되지 않음
    - 환경이 변화하여 쓸모가 없어지거나 품질이 저하될 수 있을 뿐

하드웨어 신뢰성을 보여주는 욕조 곡선(bathtub curve)과 비교됨



[그림 1-1] 욕조 곡선

- 소프트웨어 유지보수 작업은 설계의 변경을 요구함

소프트웨어의 응용 분야
응용 분야의 분류
- 시스템 소프트웨어
- 실시간 소프트웨어
  이벤트 발생과 처리가 실시간으로 이루어짐
  은행 시스템, 좌석 예약 시스템
- 내장형 소프트웨어
  대형 시스템의 일부로 내장됨
  자동차, 전자 제품 등에 내장된 제어 소프트웨어
- 비즈니스 소프트웨어
  사업 목적의 업무를 처리
  회계 업무 패키지, 재고 관리 패키지 등
- 개인용 소프트웨어
- 인공지능 소프트웨어
  복잡한 문제 해결을 위한 것
  전문가 시스템, 화상/음성 인식
- 웹 기반 소프트웨어
- 공학용/과학용 소프트웨어

1.2 소프트웨어 위기 현상

소프트웨어 위기(software crisis)
- 1968년 NATO 소프트웨어 공학 컨퍼런스
- 하드웨어 기술의 급격한 발전과 문제의 복잡화에 비해 소프트웨어의 기술 발전이 더딤을 일컫는 용어

소프트웨어 위기 현상의 사례

- 개발 일정의 지연
- 초과 비용의 발생
- 제품 신뢰도의 결여
- 명세를 충족하지 못하는 제품
- 제품의 품질 저하와 유지보수의 어려움

소프트웨어 위기 현상의 원인
- 소프트웨어 엔지니어의 부족
- 경영층의 인식 부족
- 방법론과 도구의 부재, 개발 생산성의 저하
- 소프트웨어 자체의 복잡성 증가

1.3 소프트웨어 공학

소프트웨어 공학의 유래
- 1968년 NATO 소프트웨어 공학 컨퍼런스
- 소프트웨어 위기 현상을 부각하고 해결책으로 표현하기 위해 만든 용어
- 이후로 고품질 소프트웨어의 경제적이고 빠른 생산과 유지보수를 위한 연구 분야가 됨

소프트웨어 공학의 정의
- NATO 컨퍼런스에서 바우어 교수의 정의
  "신뢰성 있고 요구기능을 효율적으로 수행하는 소프트웨어를 경제적으로 생산하기 위해 건전한 공학적 원리와 방법을 만들고 사용하는 것"
- IEEE 소프트웨어 공학 표준 용어집
  "소프트웨어의 개발, 운영, 유지보수에 체계적이고 숙달되고 정량화된 접근 방법을 적용하는 것, 즉 소프트웨어에 공학 기술을 적용하는 것"
- 기타
  "인간에게 유용한 소프트웨어 제품을 만드는 과정에 과학적 지식을 적용함으로써 실제적 문제의 비용 효율적 해결책을 다루는 일"

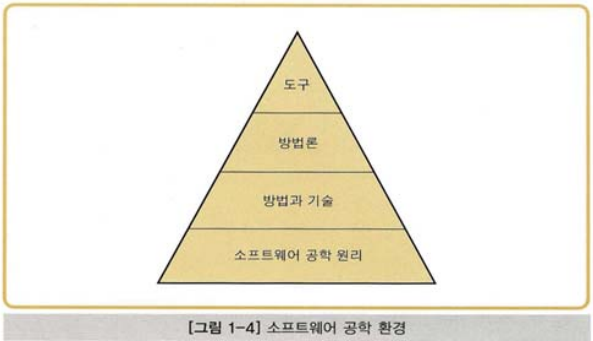
1.4 소프트웨어 공학 환경

고려 사항
- 소프트웨어 개발이 단순한 코드의 생성이 아님
- 비용을 줄이고 제품의 품질을 높이기 위한 방법은?
  문서화와 프로젝트 관리가 중요함
  초기 요구사항 명세화 작업에 노력을 기울임
  변경이나 재사용을 염두에 둔 작업이 필요함

소프트웨어 공학 환경
- 다양한 해결 방법들을 통합적으로 다루어야 함

- 환경의 구성

  소프트웨어 공학 원리에 기초하여 방법과 기술이, 방법과 기술을 가지고 방법론이, 방법론에 기초하여 도구가 만들어짐



[그림 1-4] 소프트웨어 공학 환경

- 소프트웨어 공학 원리(principle)

  소프트웨어 프로세스와 제품의 바람직한 측면들을 기술하는 일반적이고 추상적인 설명

  추상화, 분할정복, 계층적 조직의 원리

- 방법과 기술(method and technique)

  행위를 통제하는 체계적이고 일반적인 가이드라인

  바람직한 속성을 프로세스나 제품에 포함시키기 위해 필요

- 방법론(methodology)

  방법과 기술들의 조합으로 문제 해결을 위해 정해진 프로세스 안에서 조직화 한 것

  절차(what)와 방법(how)을 함께 기술한 것

1.5 소프트웨어 프로세스 모델

소프트웨어 프로세스 모델(process model)

- 소프트웨어의 생산과 진화 과정을 추상화하여 특정 시각에서 표현한 것
- 시스템 개념화, 요구사항 정의, 설계와 구현에 이르는 전이 과정을 표현
- 좋은 프로세스 모델은 전이 과정에서 생기는 문제를 최소화해야 함

  공통 개발 프레임워크의 제공하여 생산성을 향상시킴

  개발자 간 공통의 문화와 공통의 기술을 제공

1.6 좋은 소프트웨어의 기준

외부 품질과 내부 품질

- 외부 품질: 사용성이나 신뢰도와 같이 사용자가 인지할 수 있는 것
- 내부 품질

  외부 품질 향상에 도움을 주는 것

  잘 작성된 요구사항이나 설계 문서

소프트웨어의 신뢰도(reliability)
- 사용자가 소프트웨어를 신뢰하는 정도
- 오랜 시간 작동되며 치명적 오류가 없으며 오류가 발생 후에 무난히 복구되며 강건해야 함
- 시스템의 종류나 특성에 따라 측정 방법이 다름
    주어진 시간 간격 동안 정확히 작동될 확률, MTTF 등
- 시간이 지남에 따라 안정화되나, 소프트웨어 고장 함수가 욕조 곡선과 유사한 형태를 가지는 경우도 있음
    버그 수정이나 기능의 추가로 인해 새로운 오류가 유입되는 경우
    하드웨어나 운영체제를 변경하는 경우
    적정 용량이나 성능을 초과하여 사용하는 경우

소프트웨어의 정확성(correctness)
- 작은 결함이 있어도 정확한 것은 아니나 신뢰도에 문제가 되지 않을 수 있음

소프트웨어의 성능(performance)
- 지정된 시간 안에 컴퓨터 시스템이 처리할 수 있는 작업량

소프트웨어의 사용성(usability)
- 본래의 목적으로 효율성 있게 사용할 수 있는가의 정도

소프트웨어의 상호운영성(interoperability)
-다른 시스템과 공존하며 협력할 수 있는 능력

소프트웨어의 유지보수성(maintainability)
- 소프트웨어의 변경이 용이한 정도
- 요지보수 요인
    새로운 기능의 추가, 기존 기능의 개선, 환경의 변화에 따른 수정, 존재하는 오류의 수정

소프트웨어의 이식성(portability)
- 다른 환경에서 동작할 수 있는 능력
- 이식 요인
    하드웨어, 운영체제, 상호작용하는 시스템의 변경 등

소프트웨어의 검사성(verifiability)
- 좋은 소프트웨어의 속성을 갖추었는지 검사할 수 있는가

소프트웨어의 추적성(traceability)
- 관련자(stakeholder), 요구사항, 설계 문서, 소스 코드 간의 관계를 추적할 수 있는가

메뉴	**연습문제**

1. 소프트웨어 위기 현상을 가져온 원인에 해당되지 않는 것은?
   ① 소프트웨어 규모와 복잡도의 증가
   ② 프로젝트 관리 기술의 부재
   ③ 소프트웨어 엔지니어의 부족
   ④ 소프트웨어 수요의 감소

<정답> ④
<해설> 하드웨어 성능이 향상되면서 소프트웨어 수요는 증가하고 요구사항은 복잡해진다.

2. 소프트웨어의 특성이 아닌 것은?
   ① 물리적 마모에 의해 사용할 수 없게 된다.
   ② 유형의 매체에 저장되나 보이지 않는다.
   ③ 수학이나 물리학에서 나타나는 규칙적이고 정형적인 구조가 없다.
   ④ 사용자 요구나 환경의 변화에 적응될 수 있다.

<정답> ①
<해설> 소프트웨어는 물질적 특성을 가지지 않는다.

3. 소프트웨어 공학의 목표에 해당하지 않는 것은?
   ① 품질 좋은 소프트웨어
   ② 소프트웨어의 경제적 생산
   ③ 자원을 최대한 사용
   ④ 계획된 일정에 맞게 소프트웨어를 생산

<정답> ③
<해설> 자원을 최소한으로 효율적으로 이용해야 함

4. 소프트웨어 개발 프로세스(what)와 개발 방법(how)을 결합한 개념은?
   ① 기술(technique)　　② 소프트웨어 개발 방법론(methodology)
   ③ 소프트웨어 생명 주기(life cycle)　　④ 소프트웨어 공학 원리(principle)

<정답> ②
<해설> 소프트웨어 생명주기는 탄생에서 폐기까지의 변화 과정을 의미하는 개념이며 기술은 방법에 비해 기계적이며 제한적인 적용성을 가지는 용어

5. 소프트웨어 품질 목표 중 소프트웨어를 다른 환경으로 이식할 경우에도 쉽게 수정될 수 있는 능력을 의미하는 것은?

① 성능 ② 추적성
③ 이식성 ④ 효율성

<정답> ③
<해설> 환경이란 기존과 다른 하드웨어, 운영체제 또는 상호작용 하는 다른 시스템을 말한다.

메뉴	**정리하기**

1. 포괄적 의미의 소프트웨어란 무엇인가?

□ 소스 코드와 데이터는 물론 모든 관련 문서들까지 포함하는 개념

2. 소프트웨어 위기 현상이란 무엇인가?

□ 하드웨어 발전에 비해 소프트웨어 발전이 더딤을 나타내기 위한 용어로 소프트웨어 공학 기술의 후진성을 의미

3. 소프트웨어 공학의 정의는 무엇인가?

□ 소프트웨어의 개발, 운영, 유지보수에 체계적이고 숙달되고 정량화된 접근 방법을 적용하는 것, 즉 소프트웨어 개발 과정에 공학 기술을 적용하는 것

4. 소프트웨어 개발 방법론이란 무엇인가?

□ 문제 해결을 위해 필요한 여러 방법과 기술들이 정해진 프로세스 안에서 함께 묶인 것이다. 프로세스(what)와 방법(how)을 포함한 개념

5. 좋은 소프트웨어인지 판단하기 위한 기준으로 무엇이 있는가?

□ 신뢰도, 정확성, 성능, 사용성, 상호운영성, 유지보수성, 이식성, 검사성, 추적성 등