소프트웨어 공학

Lecture #0: 강좌 소개

Eun Man Choi
emchoi@dgu.ac.kr

목 차

- 소개
- 강좌의 목표와 내용
- 강의 진행 및 평가 방법
  - 스케줄
  - 평가 방법
  - 프로젝트
- 소프트웨어 엔지니어링이란?

소개

- 담당 교수
  - 최은만
  - http://eclass.dongguk.edu
  - softengineering@groups.facebook.com
  - emchoi@dongguk.edu
  - 정보문화관 Q306

- 예상하는 수강자
  - 프로그래밍 경험자
  - 실무 소프트웨어 개발 방법을 배우고 싶은 사람
  - 협업을 통하여 소프트웨어 개발을 경험하고 싶은 사람
  - …

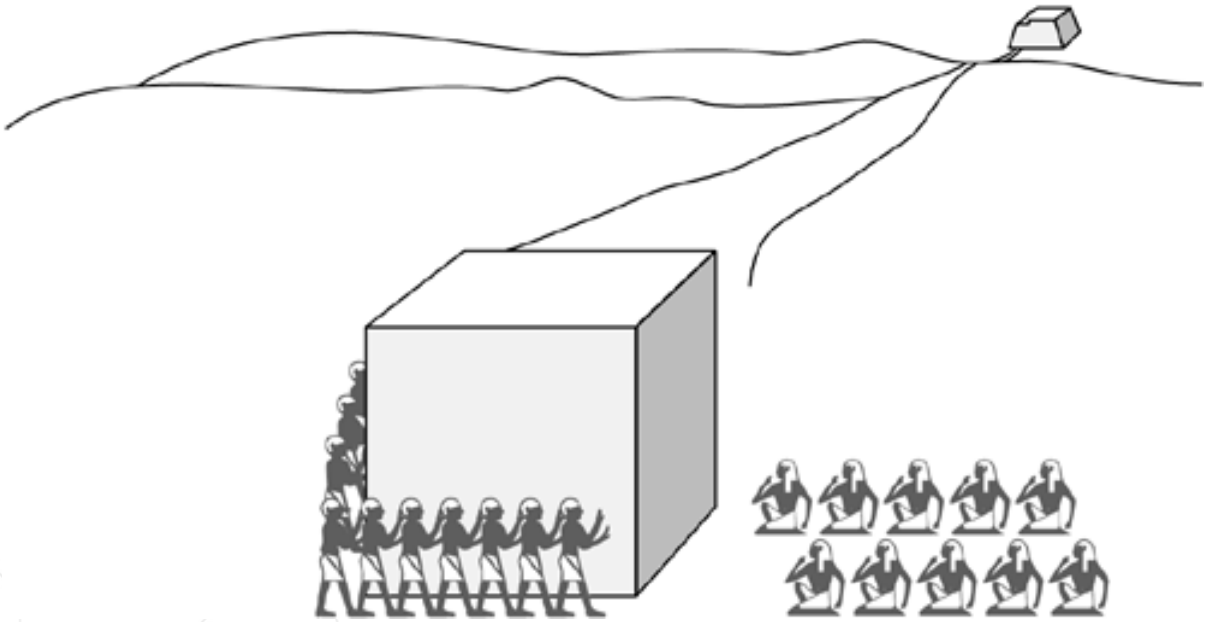
이 강좌의 목표

1. 소프트웨어 개발 기술의 습득
   - 요구 분석 방법
   - 설계 기법
   - 코딩 스타일, 테스트 기법
   - 릴리스 및 유지보수 기술
   - 객체지향 설계 기술(UML)

2. 실용 소프트웨어를 개발하기 위한 프로젝트 경험
   - 공동의 목표를 위한 효과적인 커뮤니케이션 방법
   - 아이디어와 진행 상황을 분명히 표현할 수 있는 능력 개발
   - 문제점을 파악하는 능력과 타협 기술

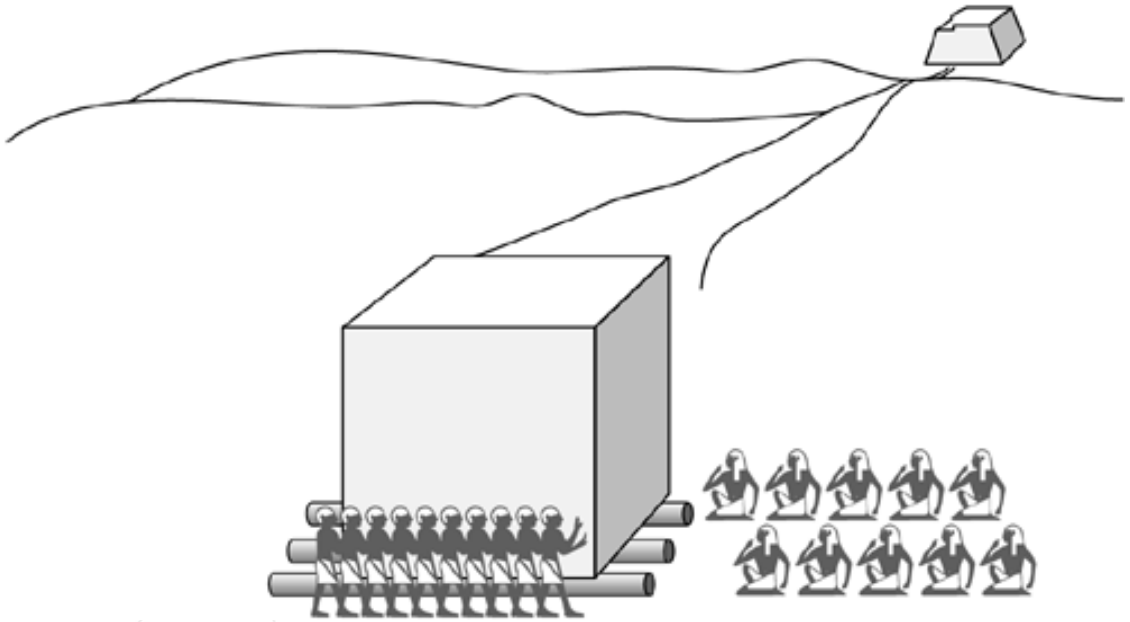
Programmer → *Software Engineer*

강의 내용 비유



- 강에서 10K 떨어진 곳에
- 20명이
- 100일 동안 피라미드 만들기

강의 내용 비유



- **Smart team**
  - Go slow to go FAST
- Code-and-Fix 식 개발(X)
- Engineering 식 접근 방법(O)

강의 내용 비유

- 규모에 따라
  - 계획, 설계, 만드는 방법, 전략이 달라짐

강의의 핵심

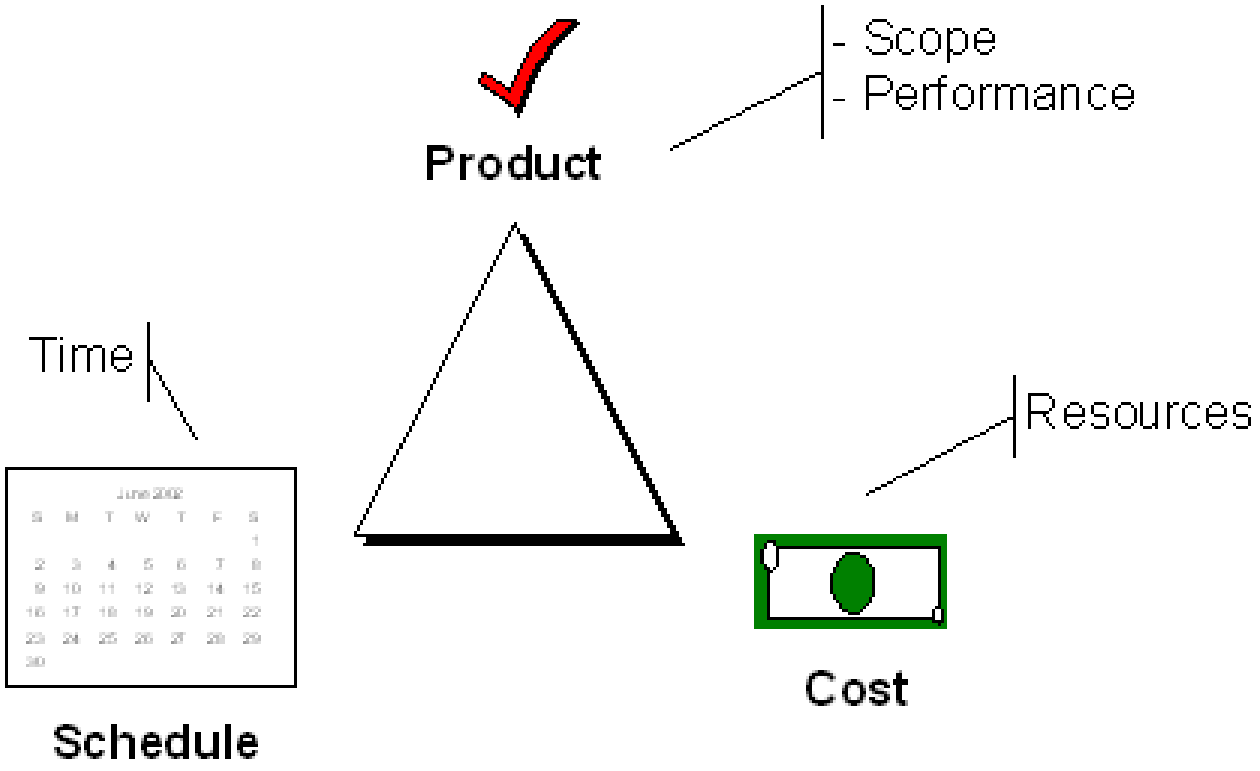
- 소프트웨어 개발에 엔지니어링 식의 접근 방법 적용
- 주먹구구가 아니라 효율적인 방법을 찾아보고 밝혀진 원리를 적용
  - 절차적인 프로그래밍의 설계 방법 – '구조적 방법'
  - 객체지향 프로그래밍의 설계 방법 – '객체지향 방법'
- 개발 전 단계에 대한 Best Practice 소개
  - 계획, 요구분석, 설계, 코딩, 테스트, 유지보수, 형상 관리 작업 방법
  - 관련 도구

소프트웨어 개발의 핵심 변수

- 소프트웨어 개발의 성패에는 다음과 같은 사항이 관련되어 있다.
  - 아이디어를 어떤 쓸모 있는 결과물로 바꾸는 과정(프로세스)
  - 제한된 시간과 자원
  - 고객의 만족
  - 리스크를 관리
  - 개발 인력의 경험 및 팀워크(커뮤니케이션)

소프트웨어 프로젝트 트라이앵글

- 프로젝트는 성공적인 소프트웨어라는 결과를 내기 위하여 세 가지 요소의 균형을 유지하여야 한다.



강의 방법

- **Active Learning을 위한 Inverted Class**
  - 소프트웨어를 협력하여 개발하기 위한 Best practice의 소개와 경험을 통한 학습

1. **In-class 액티비티**
   - 팀 조직, 팀 빌딩, 요구를 찾기 위한 게임스토밍
   - 디자인 스튜디오, 도구 활용 실습 및 설계, 인스펙션 회의
   - 테스트 케이스 개발 워크샵
2. **Out-of-class 학습**
   - 동영상을 보며 원리 및 방법 학습
   - 교과서를 미리 읽어 원리 학습
   - 코딩

바람직한 강의

주제에 대한
지식
- 저자 직강

교수-학생
인터랙션
- Facebook

교수 방법
- Inverted
Class

강좌 관리
- e-Class

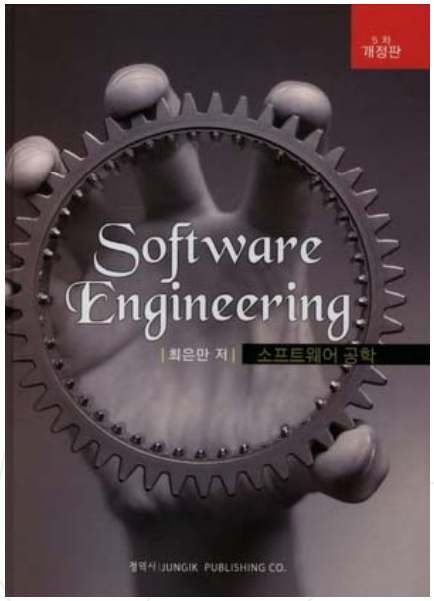
성적 평가 방법

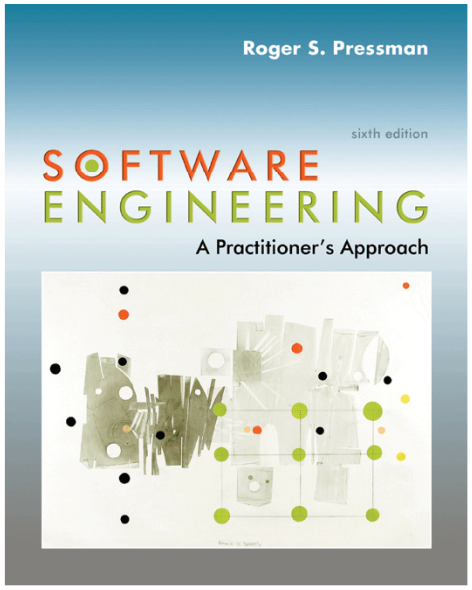
60%	중간시험(30%) 10월 22일(토) 오전 10시 기말시험(30%) 12월 10일(토) 오전 10시
30%	프로젝트 제안(2%) 요구 분석(7) 설계서(7) 베타 릴리스(7) 최종 릴리스(7)
10%	출석과 수업 참여도 (3회 지각은 1회 결석으로 간주 학업세칙 47조, 1/5 이상 결석은 F 학업세칙 46조)

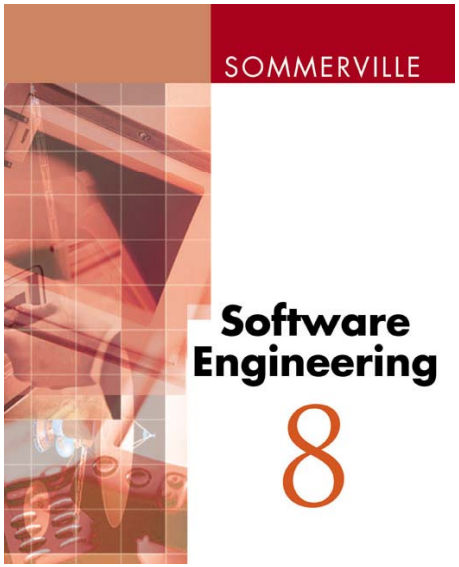
- 상대 평가와 절대 평가를 혼용
  - 100점 만점 중 40점 미만은 F
  - 40점 이상은 상대 평가
- 팀 프로젝트는 팀 단위로 평가

강의 교재 및 참고문헌

- 강의 교재
  - 최은만, 소프트웨어 공학(5차 개정판), 정익사. 2011.







- 참고 문헌
  - Pressman, Software Engineering(7th edition), 2011.
  - Sommerville, Software Engineering(8th edition), 2008.

강의 주제

- 소프트웨어 프로세스
- 계획
- 요구 분석
  - 구조적 분석
  - 사용 사례
- 설계
  - 설계 원리
  - 구조적 설계
  - 객체지향 분석 및 설계
    - Unified Modeling Language
  - 모듈 설계와 UI 설계
- 코딩
- 테스트
- 유지보수

프로젝트

- 프로젝트 팀 구성
  - 다음 주 액티비티
  - 주제부터 모든 것 스스로 정함
- 4명의 학생으로 팀 구성
  - 실제 프로젝트와 같이 팀 구성(리더, Writer, 엔지니어,...)
  - 정확한 역할 분담과 팀 스피릿 발휘
- 단계별 결과
  - 제공한 템플릿을 이용한 결과물 생성
- 평가 요소
  - 내용, 완성도, 스타일(포맷)

강의 일정(전반부)

주	날짜(요일)	Active Learning(동영상 강의)	Activity In Class	과제 부 여	마감
1	9/1(목)		L00 강의 소개 소프트웨어 공학이란 무엇인가?		
	9/6(화)	L01 개발 프로세스	A01 프로젝트 팀 조직 개발 프로세스 케이스 스터디		
2	9/8(목)	L02 요구분석 -요구란?, 요구추출과 분석	A02 팀 빌딩 ● 프로젝트 주제 논의	프로젝트 문제 제안	
	9/13(화)	추석 휴일			
3	9/15(목)	L03 구조적 분석 -DFD, 자료 사전, Mini-spec	A03 기능 리스트 작성	요구분석명세서(SRS) -양식 참조	프로젝트 문제 제안
	9/20(화)	L04 객체지향 개념 -객체지향 특징, 기본개념, UML, 클래스 관계	A04 객체와 클래스 찾기		
4	9/22(목)	L05 사용 사례 -개념, 작성 방법, Exercise	A05 사용 사례 다이어그램 및 기술		
	9/27(화)	L06 클래스 다이어그램 - 클래스, 객체, 속성, 오퍼레이션, 관계	A06 StarUML 사용법 강의 ● 클래스 다이어그램 그리기		요구분석명세서(SRS)
5	9/29(목)	L07 인터랙션 다이어그램 - 순서 다이어그램, 커뮤니케이션 다이어그램	A07 순서 다이어그램 그리기		
	10/4(화)	L08 설계원리 - 추상화, 모듈화, 구조적 설계	A08 설계 원리 찾기. 참고 자료(Code complete 2)	설계서(SD)	
6	10/6(목)	L09 아키텍처 - 시스템 요소, 아키텍처 스타일	A09 아키텍처 설계		
	10/11(화)	L10 UI 설계 - UI 설계 요소, 나쁜 설계 사례	A10 UI 설계		
7	10/13(목)	L11 설계 패턴	A11 설계 패턴 마이닝		
	10/18(화)	L12 구현(1) - 코딩 원리, 코딩 스타일	A12 프로젝트 코딩 표준 만들기		
8	10/20(목)	중간시험 대비 리뷰 및 질의응답	범위: 1, 3~6 장		설계서(SD)
9	10/29(토)	10:10 AM 중간시험 장소: 미정			

강의 일정(후반부)

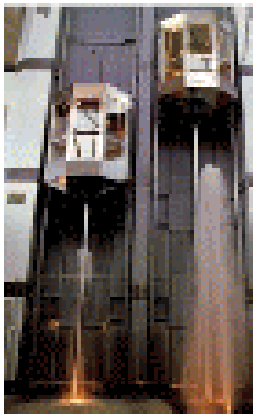
10	11/1(화)	L12 구현(2) - 리팩토링, 인스펙션	프로젝트 코딩	코딩(베타 릴리스)	
	11/3(목)	L13 UML 코딩 -정적 동적 다이어그램의 매핑	A13 UML 코딩		
11	11/8(화)	L14 테스트(1) - 원리, 화이트 박스 테스트	A14 블랙박스 테스트 케이스 만들기		
	11/10(목)	L14 테스트(2) - 블랙 박스 테스트, 객체지향 테스트, 통합 테스트, 인수 테스트	A14 화이트박스 테스트 케이스 만들기		
12	11/15(화)	L15 유지보수	A15 Bugzilla와 Subversion 튜토리얼		
	11/17(목)	L16 계획(1) - 범위설정, 일정 계획	A16 MS Project 튜토리얼		
13	11/22(화)	L17 계획(2) - 노력 추정, 조직 계획, 위험 분석			
	11/24(목)	L18 품질 보증(1) - 개념, 활동			
14	11/29(화)	L19 품질 보증(2) - 프로세스 품질, 프로덕트 품질, 인스펙션			코딩(베타 릴리스) 및 팜플렛 원고
	12/1(목)	최종 데모 Part 1	최종 데모(1)		
15	12/6(화)	최종 데모 Part 2	최종 데모(2)		
	12/8(목)	기말 시험 샘플 풀이	기말 시럼 리뷰	코딩(최종 릴리스)	
16	12/10(토)	10:00 AM 기말 시험	범위: 2장, 7~10장		최종 릴리스(12/12)

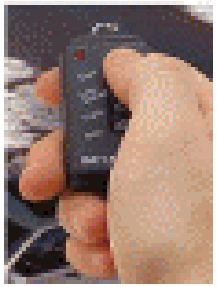
클래스 웹 사이트

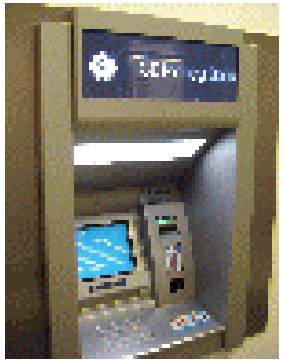
- 웹
  - http://eclass.dongguk.edu/
- 리소스
  - 강의 슬라이드
  - 액티비티 워크북
  - 프로젝트 문서양식
- 제출
  - 프로젝트 과제는 리더만 e-Class에 제출할 것

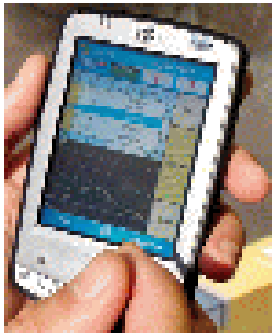
소프트웨어와 우리 생활

- 의존성(dependability)








소프트웨어가 비즈니스를 주도

- 소프트웨어 업체










- 기존 업체









소프트웨어 엔지니어

[표 Ⅰ-3] 미국의 10대 좋은 직업(2006)

(단위: %, $)

순위	직업명	고용성장률	평균임금
1	소프트웨어 엔지니어(Software Engineer)	46.1	80,427
2	대학교수(College Professor)	31.4	81,491
3	금융 및 재정 상담가(Financial Advisor)	25.9	122,462
4	인적자원관리자(Human Resources Manager)	23.5	73,731
5	의사보조원(Physician Assistant)	49.6	75,117
6	시장조사 분석가(Market Research Analyst)	20.2	82,317
7	컴퓨터/IT 분석가(Computer/IT Analyst)	36.1	83,427
8	부동산 감정사(Real Estate Appraiser)	22.8	66,216
9	약사(Pharmacist)	24.6	91,998
10	심리학자(Psychologist)	19.1	66,359

※ 출처: CNN(2006)

소프트웨어

- 소프트웨어
  - 프로그램 + 프로그램의 개발, 운용, 보수에 필요한 정보 일체(소프트웨어 생산 결과물 일체)
- 소프트웨어 생산
  - 소프트웨어는 프로그램의 동적인 실체
  - 프로그램은 형식 언어로 표현된 지적 노동의 결과물
  - 제조업 vs. 서비스업(소프트웨어는 제작이 아니라 창조적 노력이 포함된 개발)
  - 닳아 없어지는 것이 아니라 소용없어 못쓰게 됨
  - 논리적인 요소로 구성(유지 보수자 복잡)
  - 소프트웨어 산업(국내는 12조 규모의 산업)
- 사회적, 경제적인 소프트웨어 의존도 커짐
  - dependability

소프트웨어의 특성

- ~ilities
  - 비가시성(Invisibility)
  - 테스트 가능(Testability)
  - 복잡성(Complexity)
  - 변경성(Changeability)
  - 순응성(Conformity)
  - 장수(Longevity)
  - 복제 가능(Duplicability)
  - 응용에 의존(Application dependability)
- 제품으로서의 품질 특성
  - 좋은 소프트웨어라고 평가 할 수 있는 기준

소프트웨어 시스템

- 유기적으로 상호 작용하는 개체들의 모임
- 소프트웨어는 컴퓨터를 기반으로 하는 여러 시스템과 관계를 맺고 있음
- 특징
  - 시너지 효과
  - 역동적으로 발전, 변경
  - 상충되는 요구와 이해 관계의 절충
- 소프트웨어 자체도 하나의 시스템

정보 시스템(Information System)

- 자료의 분류, 저장, 검색에 관점
- 데이터 베이스를 대화식으로 접근
- 조직의 문제 해결을 위한 도구
- 예
  - 항공권 예약 시스템, 신용 카드 검색 서비스, 뱅킹 시스템
- 특징
  - 대규모 자료, 정적이 아님
  - 시스템 분석, 유지보수가 중요
- MIS
  - 운영, 관리, 의사결정을 위하여 정보를 제공하는 시스템

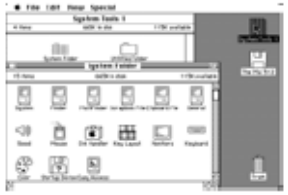
제어 시스템(Control System)

- 사건을 감지하여 처리하고 자동적으로 보고
  - 센서의 감지
  - 제어 기기의 상태 보고
  - 오퍼레이터의 입력 처리
  - 인터페이스
- 예
  - 교통 제어
  - 공정 제어
  - 수치 제어
  - 의료 시스템
  - 무기
  - 항공 제어

탑재 시스템(Embedded System)

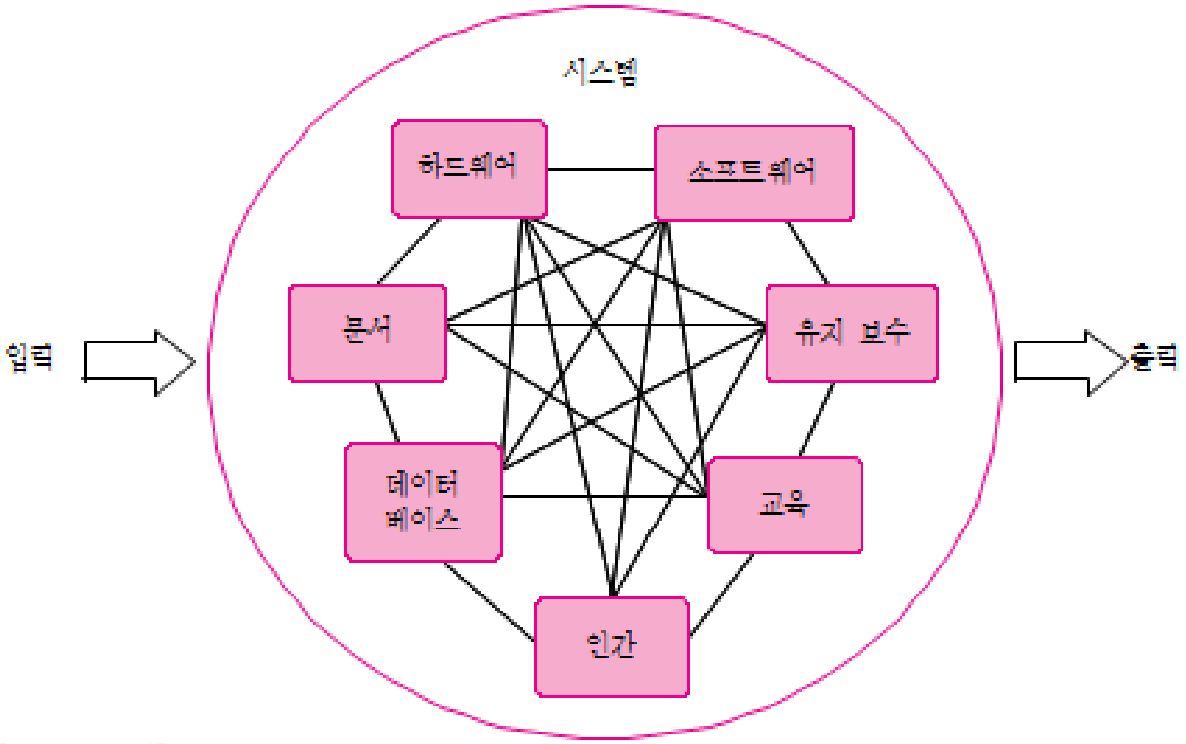
- 계산이 주된 기능이 아닌 시스템의 한 구성요소
- 예
  - 전자 기계 장치, 공정 제어
  - 비행기 유도, 스위칭 시스템
  - 환자 감시 시스템, 레이다 추적 시스템
- 특징
  - 대규모, 장기 사용, 테스트하기 어려움
  - 인터페이스가 복잡, 비동기, 병렬, 분산
  - 대규모의 자료를 접근, 변경, 출력
  - 실시간 제어, 인터페이스
  - 엄격한 요구: 실시간 반응, 고장에 대한 안전, 신뢰성

소프트웨어의 유형

소프트웨어 분류		특 징
응용 소프트웨어		• 비즈니스 업무 등 한 회사 또는 기관의 내부에서 사용하는 시스템 • 급여 시스템, 회계 시스템, 재고관리 시스템, 수강신청 등 학사업무 시스템, 은행 시스템
시스템 소프트웨어		• 운영체제, 장치 드라이버, 컴파일러, 코드 라이브러리
주문형 소프트웨어		• 특정 고객 또는 기업의 요구를 만족시키기 위하여 제작한 소프트웨어
패키지 소프트웨어		• 패키지화 하여 상업적으로 판매하는 소프트웨어 • 워드프로세서, 스프레드시트, 유통업체의 POS(Point of Sales) 시스템, 재정 분석, 주문 관리, 회계 관리 시스템
임베디드 소프트웨어		• 다른 시스템에 내장된 소프트웨어

시스템

- 네 가지 중요한 성질
  - 서브 시스템
  - 기능적 분할
  - 시스템 경계
  - 자동화 경계

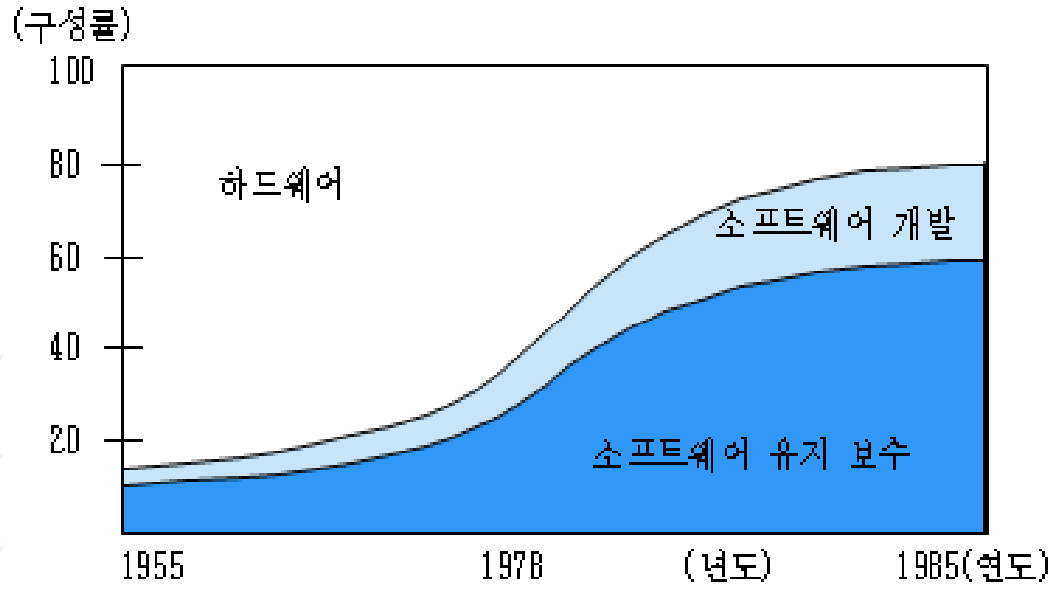


소프트웨어 공학이 다루는 문제

- 제품으로 만든 소프트웨어
  - 견고한(industrial strength) 소프트웨어

1) 고비용

- LOC로 계산하는 소프트웨어 비용 사례
  - 5만줄의 프로그램 – 4천만원 내지 1억 2천 정도의 비용
- 1억원의 소프트웨어가 1천만원 정도의 하드웨어에서 실행됨

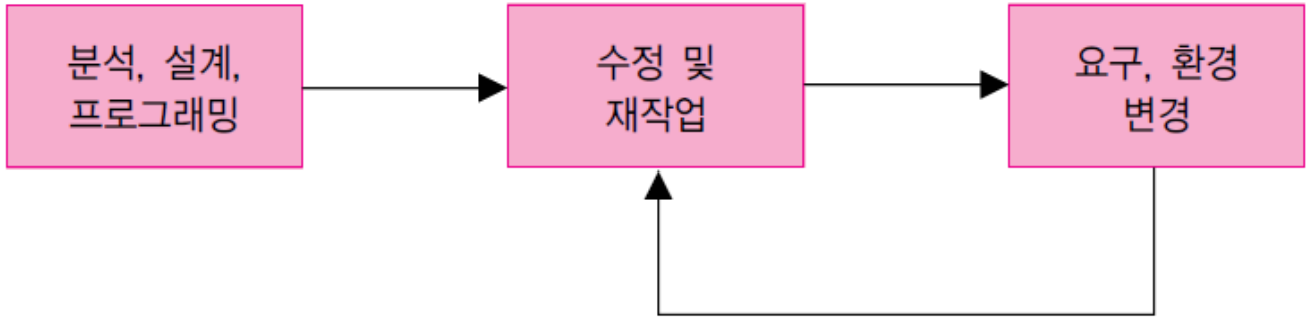


2) 지연과 낮은 신뢰도

- 계획에서 벗어난 컴퓨터 관련 개발 프로젝트
  - 600여 회사를 조사하였더니 35% 이상
- 예상대로 작동하지 않는 사례
  - 방위산업 보고(70% 이상)
- 다른 요소(하드웨어)와 다름
  - 노후화에 의한 물리적 특성의 변화에 의한 것이 아님
  - 설계, 개발 과정에 유입된 오류에 의한 것

3) 유지보수와 재작업

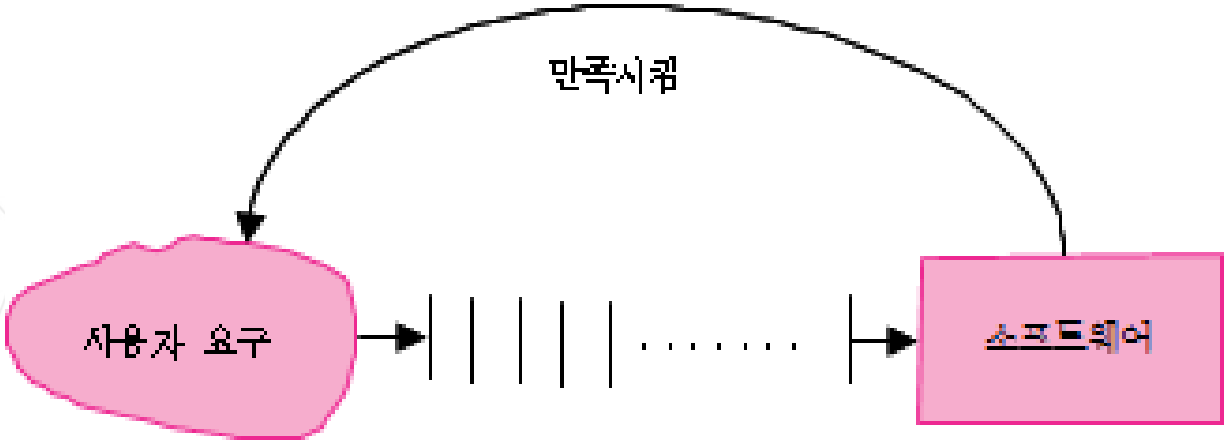
- 마모되거나 교체할 필요로 유지보수 하는 것이 아님
  - 시스템에 남아 있는 오류가 있기 때문에
- 소프트웨어는 업그레이드가 흔함
- 의도하는 바가 잘 드러나지 않음
  - 요구를 파악하기 어려움



소프트웨어 공학의 도전 과제

- 소프트웨어 공학이란?
  - 소프트웨어의 개발과 운영, 유지보수, 소멸에 대한 체계적인 접근 방법

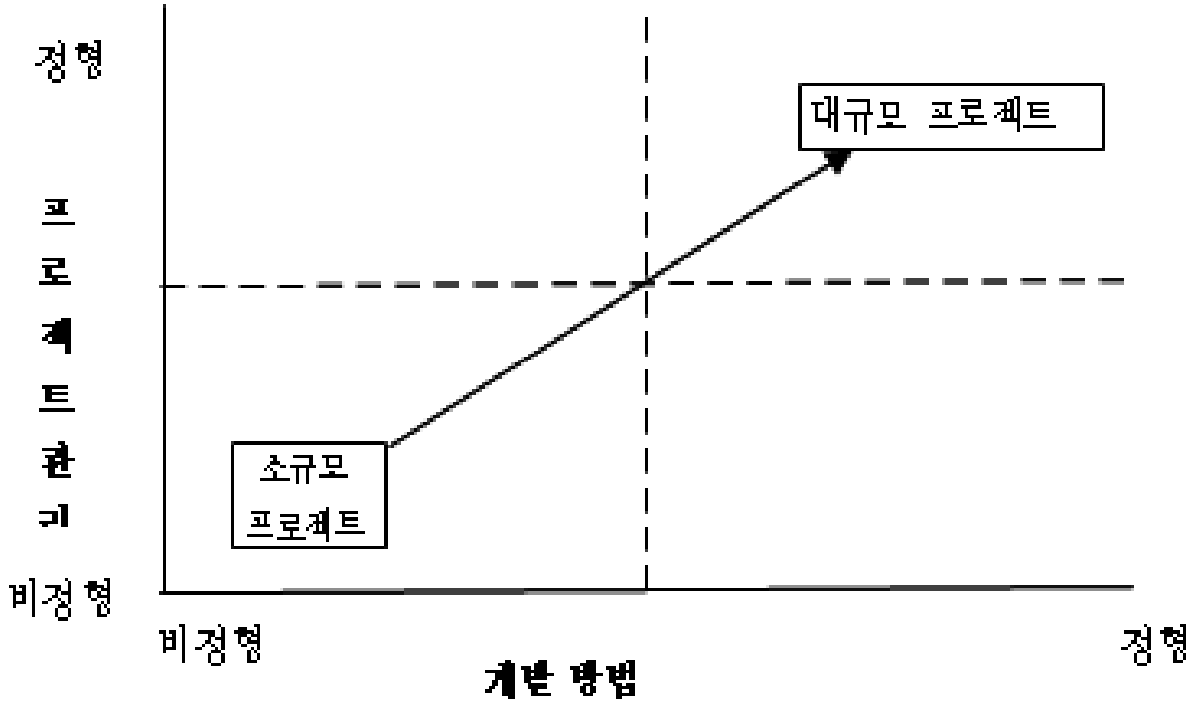
- 다루는 문제
  - 사용자의 요구를 만족시키기 위하여 소프트웨어를 체계적으로 개발하는 것



1) 규모 문제

- 수백 줄의 프로그램을 개발하는 데 사용하는 방법과는 다른 방법을 적용

- 엔지니어링 식 접근 방법 - 방법, 절차, 도구 사용



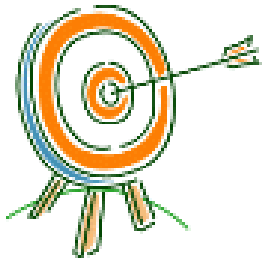
2) 품질과 생산성

- 엔지니어링 작업에서는
  - 비용, 일정, 품질과 같은 변수가 중요

- 비용
  - Man-Month로 측정

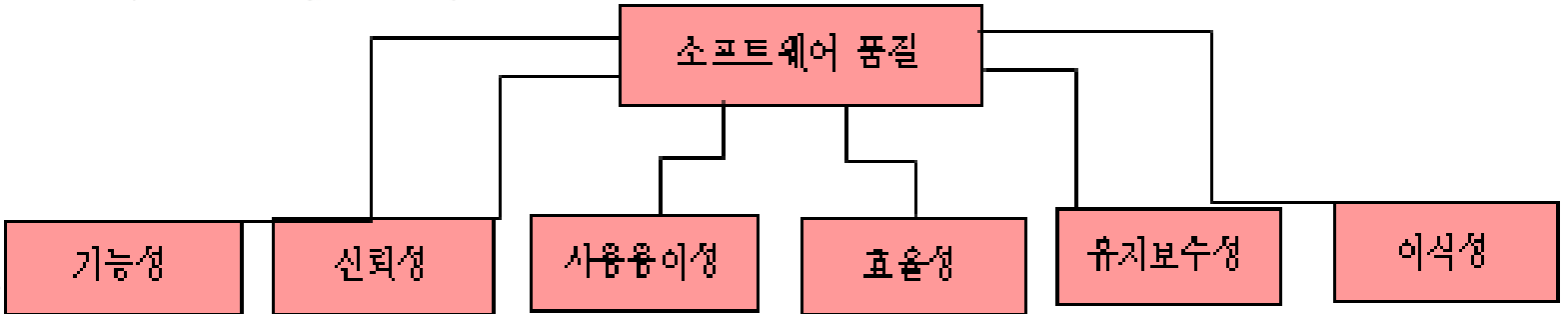
- 일정
  - 짧은 time-to-market

품질



생산성

- 품질



3) 일관성과 재현성

- 일관성
  - 프로젝트의 결과를 어느 정도 정확하게 예측가능
  - 더 높은 품질의 제품을 생산
- 프로세스의 표준화가 필요
  - ISO 9001
  - CMM(Capability Maturity Model)
- 재현성
  - 개발하는 시스템 마다 높은 품질과 생산성을 갖도록 만드는 것
  - 개발 능력, 결과의 재현성

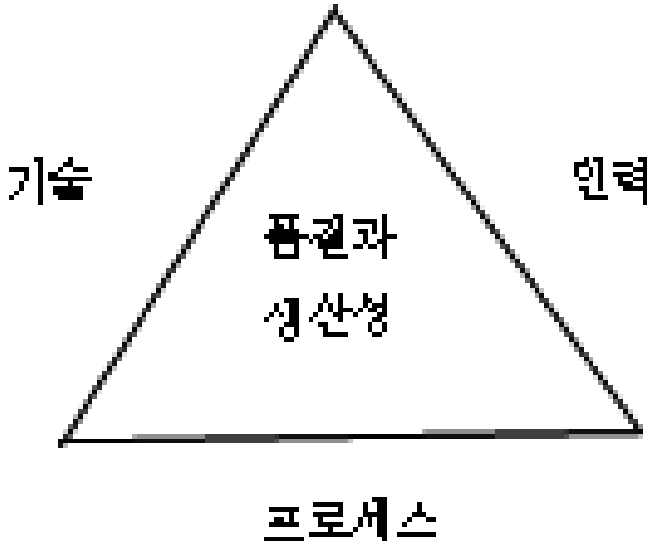
4) 변경

- 소프트웨어는 계속 진화하고 변경됨
- 비즈니스 환경 및 소프트웨어 기술도 빠르게 발전
- 변경을 조절하고 수용하는 것이 또 하나의 과제

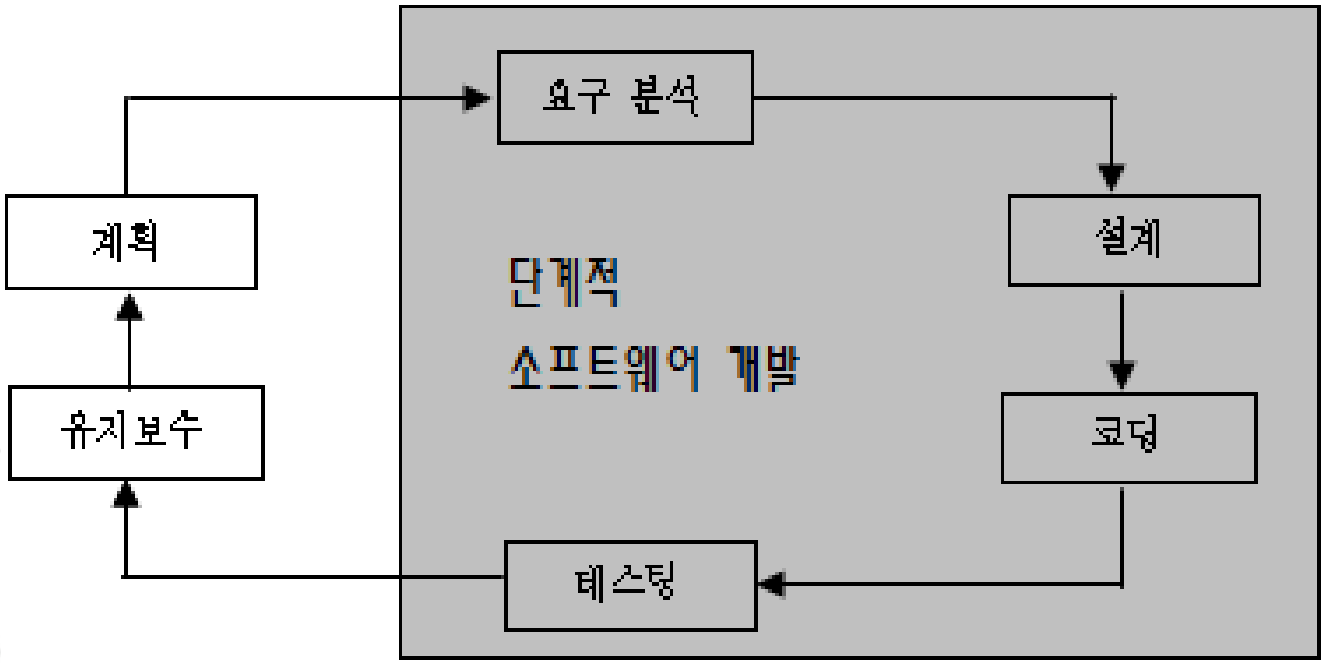
접근 방법

- 프로젝트를 수행하는 동안 얻은 품질과 생산성은 여러 가지 요인에 좌우됨

- 프로젝트 삼각 균형



1) 단계적 프로세스

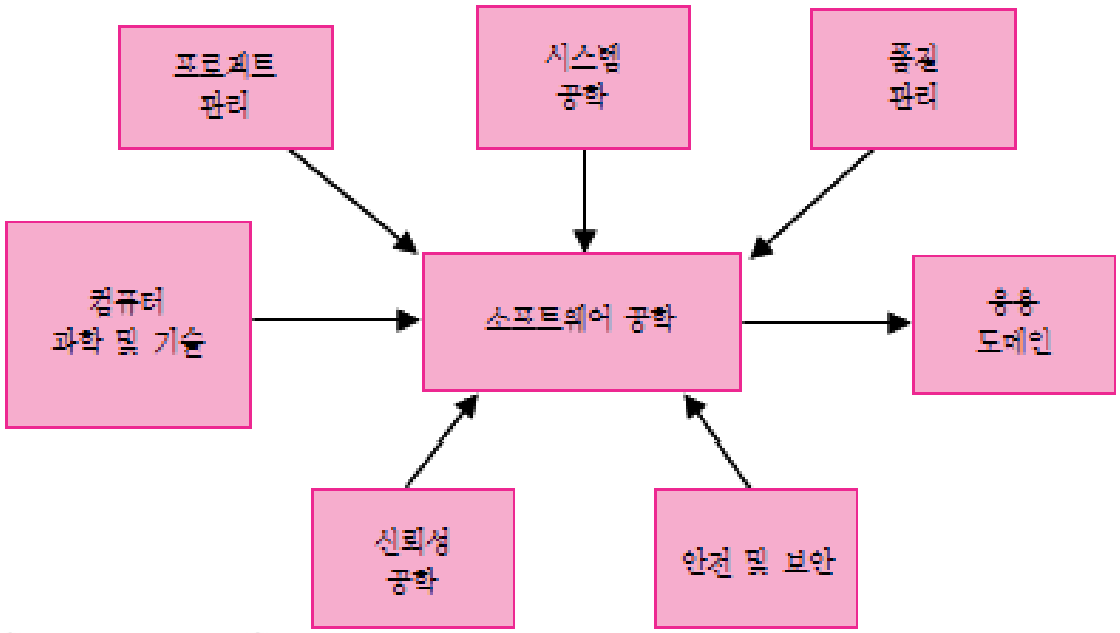


일반적인 개발 단계

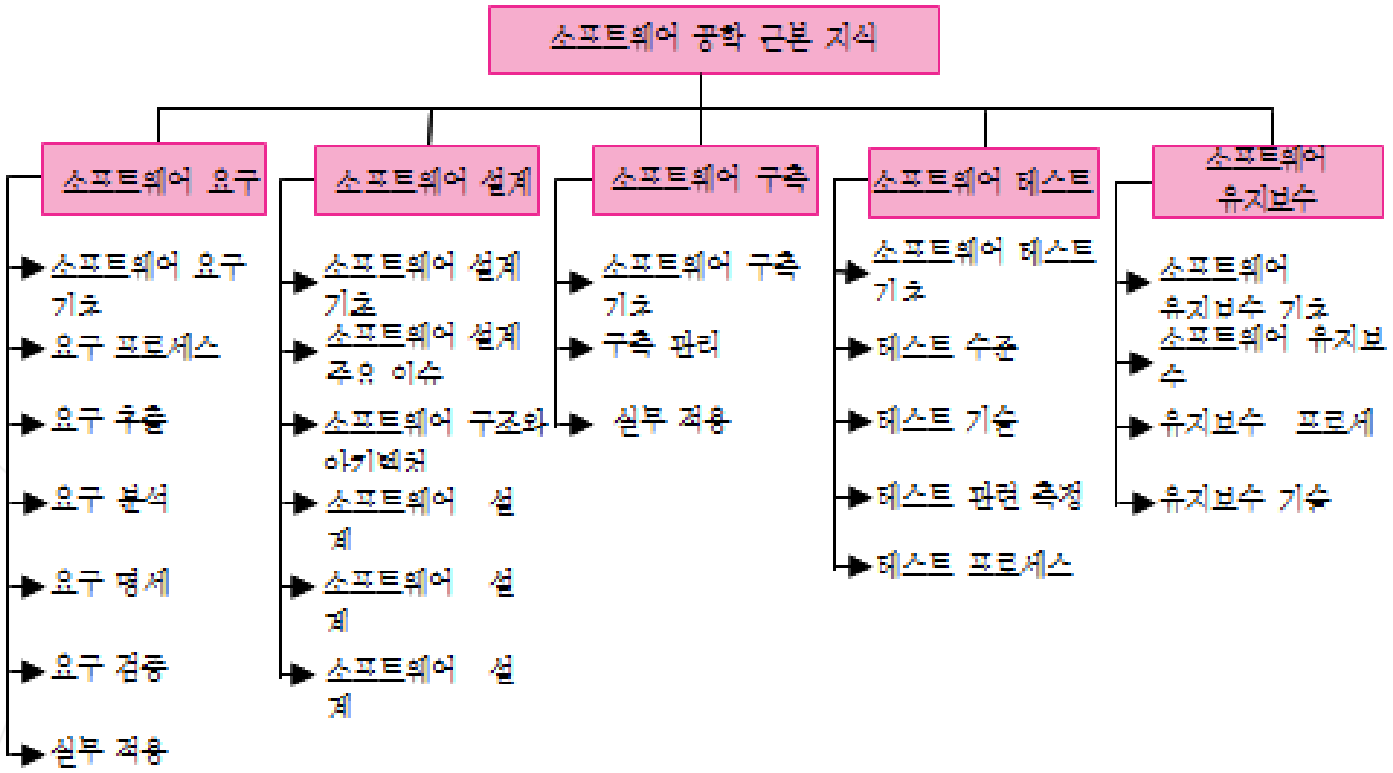
단계	초점	주요작업과 기술	결과물
분석	● 시스템을 위하여 무엇을 만들 것인가?	1. 분석 전략 수립(3장) 2. 요구 결정(3장) 3. 사용 사례 분석(4장) 4. 구조적 모델링(5장) 5. 동적 모델링(6장)	요구 명세서
설계	● 시스템을 어떻게 구축할 것인가?	1. 설계 전략 수립(7장) 2. 아키텍처 설계(7장) 3. 인터페이스 설계(9장) 4. 프로그램 설계 5. 데이터베이스, 파일 설계(8장)	설계 명세서
구현	● 시스템의 코딩과 단위 시험	1. 프로그래밍(10장) 2. 단위 테스팅(10장) 3. 시스템 안정화 및 유지보수(11장)	새 시스템, 유지보수 계획
테스팅	● 시스템이 요구에 맞게 실행되나?	1. 통합 테스팅(2장) 2. 시스템 테스팅(2장) 3. 인수 테스팅(2장) 4. 시스템의 설치(2장) 5. 프로젝트 관리 계획	테스팅 결과 보고서

근본 지식

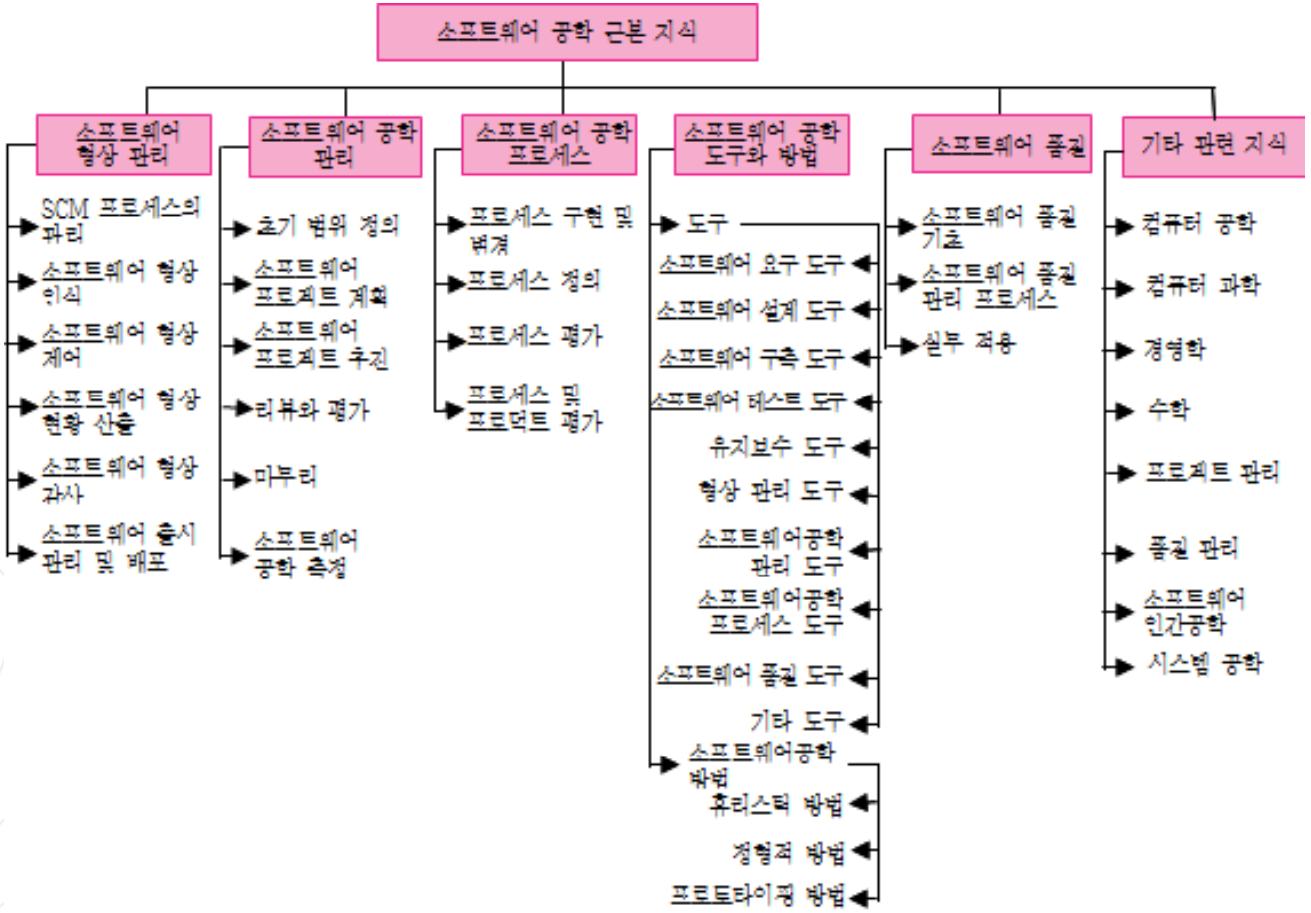
- 다른 분야와의 관계



SWEBOK



SWEBOK



Questions?

dongguk UNIVERSITY