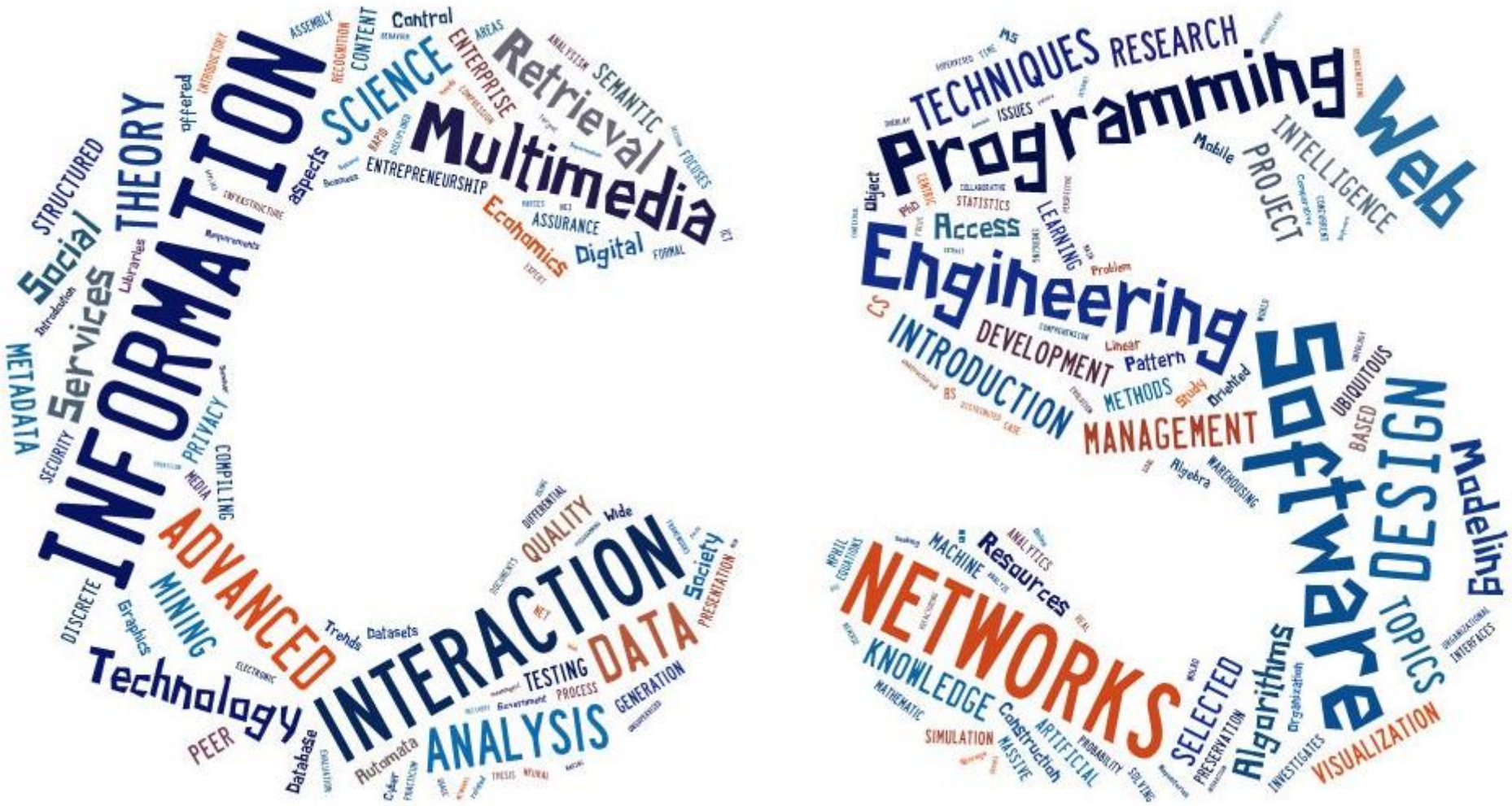

INFORMATION
SCIENCE
Retrieval
Multimedia
Control
ENTERPRISE
SEMANTIC
CONTENT
ASSEMBLY
AREAS
aspects
ENTREPRENEURSHIP
Economics
ASSURANCE
Digital
STRUCTURED
Social
THEORY
Services
METADATA
SECURITY
PRIVACY
COMPILING
ADVANCED
MINING
DISCRETE
Technology
PEER
Database
Automata
INTERACTION
QUALITY
DATA
ANALYSIS
TESTING
GENERATION
Society
PRESENTATION
Trends
Datasets
Graphics
Wide
Programming
Web
TECHNIQUES
RESEARCH
INTELLIGENCE
PROJECT
Mobile
Access
Engineering
INTRODUCTION
DEVELOPMENT
METHODS
MANAGEMENT
Software
DESIGN
Modeling
UBIQUITOUS
BASED
TOPICS
VISUALIZATION
Algorithms
SELECTED
NETWORKS
KNOWLEDGE
MACHINE
Resources
ARTIFICIAL
SIMULATION
Construction
MASSIVE
Algebra
Pattern
Linear
Problem
Object
LEARNING
STATISTICS

08 소프트웨어 공학

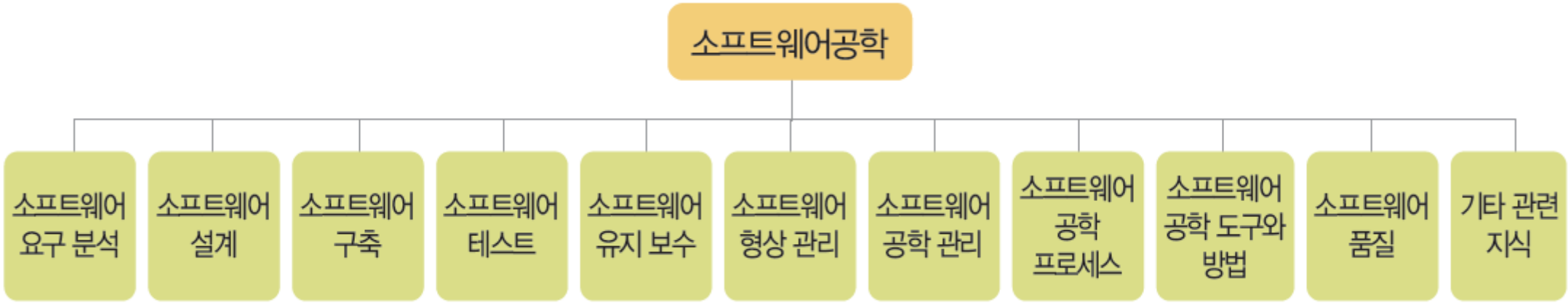
목차

공학적으로 잘 작성된 소프트웨어의 특징

- 사용자 요구사항 충족
- 높은 신뢰성
- 유지 보수의 용이성
- 쉬운 인터페이스

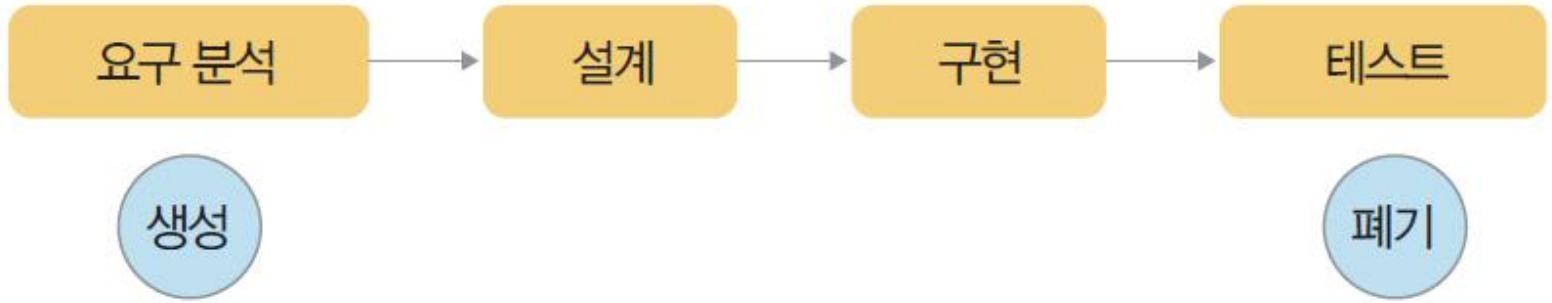
소프트웨어 위기 상황

소프트웨어 공학이 다루는 지식



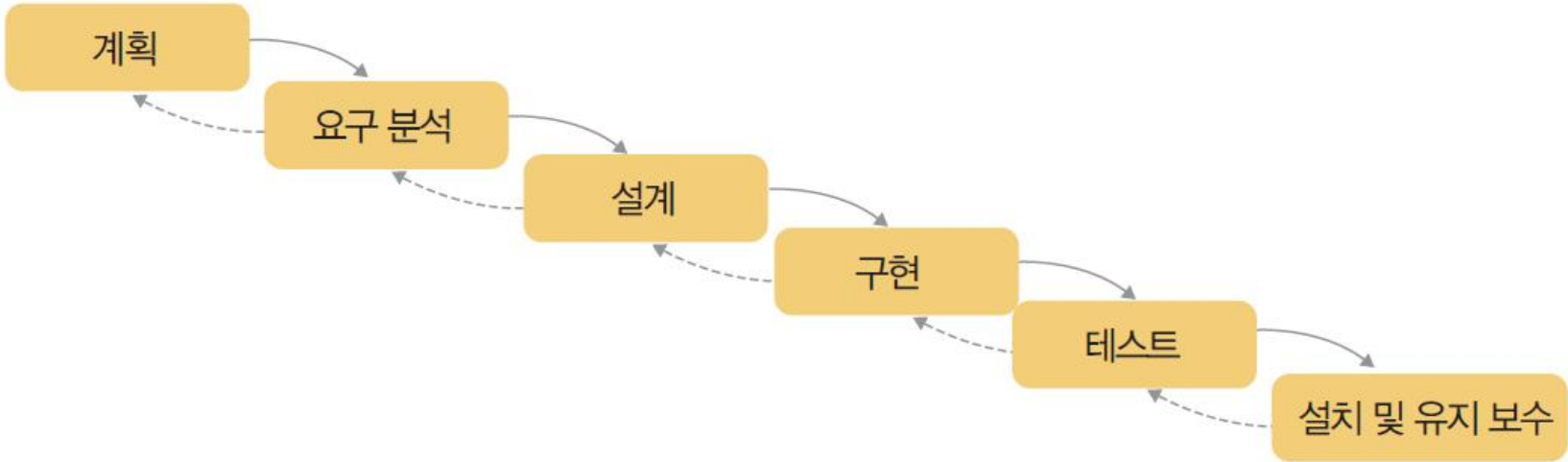
소프트웨어 개발 생명주기

- 소프트웨어 개발 생명주기(SDLC)



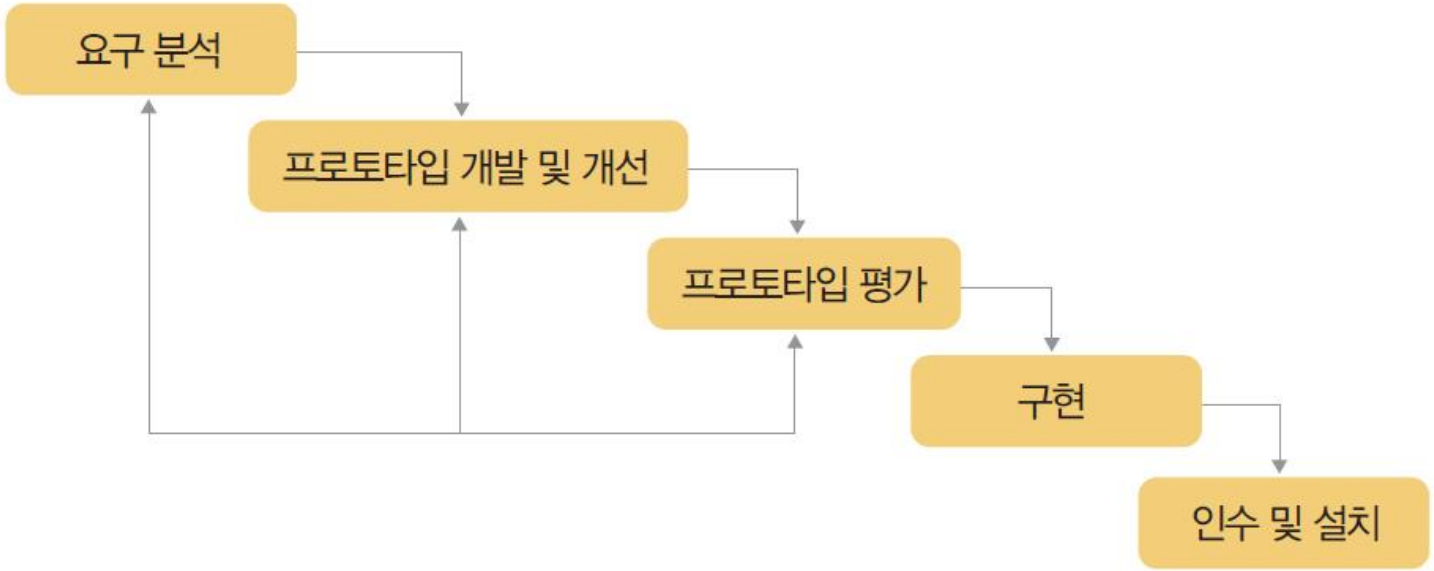
- 소프트웨어 개발 생명주기의 대표적인 모델
  - 폭포수 모델(waterfall model)
  - 프로토타입 모델(prototype model)
  - 나선형 모델(spiral model)
  - 익스트림 프로그래밍 모델(eXtreme programming model)

폭포수 모델



장점	· 응용 분야가 단순하거나 잘 알고 있는 경우에 적합하다. · 문서화가 잘 되어 있어 사용하는 데 특수 지식을 필요로 하지 않는다. · 앞 단계에 대한 결과물을 확인한 후 진행하기 때문에 안정적이다.
단점	· 요구사항이 애매할 경우 업무 분석에 과도한 시간이 필요하다. · 중간에 수정 요구가 있으면 그만큼 수정 비용이 늘어난다. · 문서를 만드는 데 과도하게 매달릴 수 있다.

프로토타입 모델



장점	· 사용자는 예상되는 결과물을 미리 보면서 수정을 요청할 수 있다. · 소프트웨어를 개발하는 데 사용자의 적극적인 참여를 유도할 수 있다. · 개발자가 사용자의 요구사항을 자세하게 알 수 있다.
단점	· 사용자가 프로토타입이 최종 산출물과 같다고 믿어 소프트웨어가 곧 완성될 거라 오해할 수 있다. · 오해를 할 경우 사용자는 전체 개발 일정을 단축하라고 요구하기 마련이고, 이는 소프트웨어 품질에 악영향을 끼칠 수 있다. · 개발자 입장에서는 중간 과정을 점검할 수 있는 산출물이 없기 때문에 프로토타입의 개발 및 개선 과정을 관리하기 어렵다.

나선형 모델

장점	· 다른 모델에 비해 완전하고 신뢰성 있는 소프트웨어를 개발할 수 있다. · 위험 요인을 사전에 분석하여 제거하거나 낮출 수 있다.
단점	· 시간과 비용이 많이 든다.

익스트림 프로그래밍 모델

- 소규모 소프트웨어 개발에 적합
- 애자일(agile) 개발 프로세스 중 하나
- 4가지 가치
  - 의사소통(communication)
  - 단순성(simplicity)
  - 피드백(feedback)
  - 용기(courage)

익스트림 프로그래밍 모델

실천 사항	내용
계획 세우기	우선순위와 기술 사항을 고려하여 개발 범위를 결정한다.
소규모 릴리즈	짧은 주기로 버전을 발표해 기술 변화에 신속히 대응한다.
상징(metaphor) 사용	전체 시스템에 추상화 개념을 적용한다. 추상화란 작업할 때 꼭 필요한 정보만 드러내고 나머지 정보는 드러내지 않는 개념이다.
단순한 디자인	사용하지 않는 기능은 추가하지 않으며, 필요 이상으로 복잡한 구조나 알고리즘을 요구하지 않는다.
지속적인 테스팅	단위 테스트를 지속적으로 수행한다. 코드를 조금씩 작성하는 방식으로 테스트를 빠르게 진행한다.
리팩토링	프로그램을 쉽게 이해하고 추후에 효과적으로 변경할 수 있도록 내부 설계 및 코딩을 수정한다.
짝(pair) 프로그래밍	두 명이 한 팀으로 프로그래밍한다. 테스트 코드를 먼저 작성하고 코드를 수정하면서 프로젝트를 진행한다.
공동 코드 소유	누구나 코드를 수정할 수 있고 책임도 함께 공유한다.
지속적인 통합	지속적으로 통합 작업을 수행한다.
주당 40시간 근무	시간외 근무를 하지 않는다. 시간외 근무를 반복하다 보면 형편없는 코드를 작성할 가능성이 높아지기 때문이다.
현장 고객 지원	고객을 개발 현장에 상주시켜 개발에 관한 의견을 수시로 교환한다.
코딩 표준	서로 동의한 코드 표준에 맞춰 코딩한다. 코드에 대한 의미 전달이 명확해지고 코드 품질에 대한 책임 공유가 보장된다.

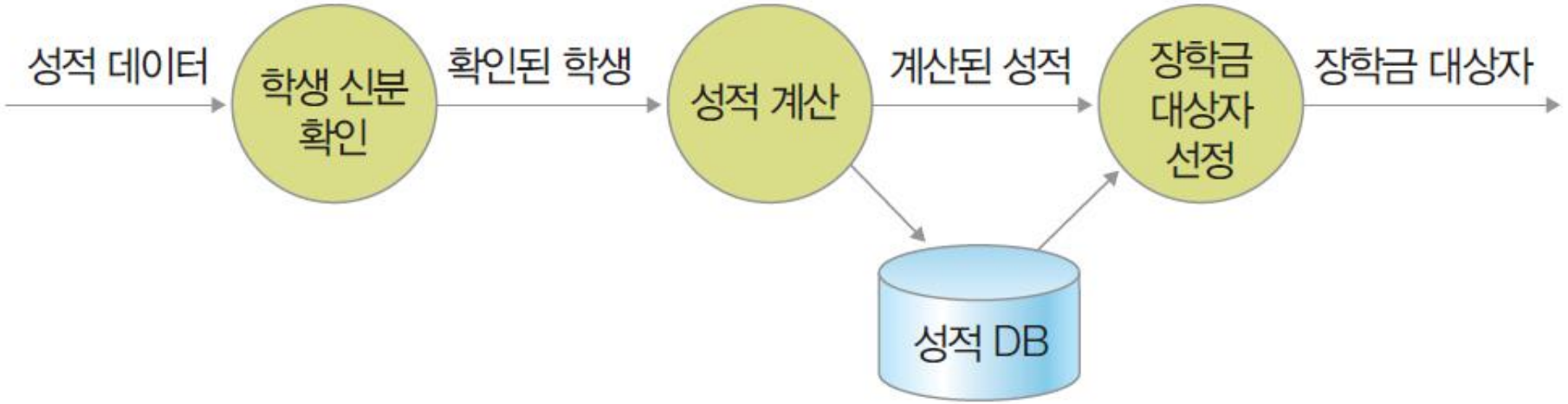
소프트웨어 개발 방법의 이해

(a) 소프트웨어 개발 생명주기 모델(이동 수단 선택)

(b) 소프트웨어 개발 방법(내비게이션, 지도, 가이드 등 구체적인 활용 방법 선택)

구조적 개발 방법

- 구조적 분석
  - 사용자의 요구를 파악하여 문서로 정리하는 단계
  - 자료 흐름도, 자료 사전, 소단위 명세서를 작성하는 순서로 진행
- 자료 흐름도
  - 소프트웨어 내부의 프로세스, 자료 저장소, 자료의 흐름을 나타내는 그래프



구조적 개발 방법

- 자료 사전
  - 자료 흐름도에서 사용된 모든 자료를 모은 것

〈정의 부분〉
성적 데이터 = 학번 + 학생 이름 + 학년 + 과목 코드 + 과목명 + [분반] + 해당 과목 성적

〈설명 부분〉
학번 = 입학 연도 2자리 + 전공 2자리 + 일련번호 4자리
과목 코드 = 10진수 4자리
분반 = 1 | 2 | 3

구조적 개발 방법

- 소단위 명세서
  - 자료 흐름도에서 작성한 프로세스의 작업 과정을 자세히 기술한 것

```
학생 데이터를 읽음
    신분이 확인된 학생 데이터가 존재하면,
        성적 데이터 파일 읽음
        과목별 성적 계산
        과목별 점수 합산
        평균 계산
        평균 출력
    신분이 확인된 학생 데이터가 존재하지 않으면,
        종료
반복
```

구조적 개발 방법

- 구조적 설계
  - 소프트웨어가 요구 사항을 기술적으로 어떻게 구현할 것인지에 초점을 둠
  - 전체 소프트웨어의 뼈대를 나타내는 구조도를 작성한 후 프로그램 설계, 자료 설계, 사용자 인터페이스 설계로 나눠 수행

구조적 개발 방법

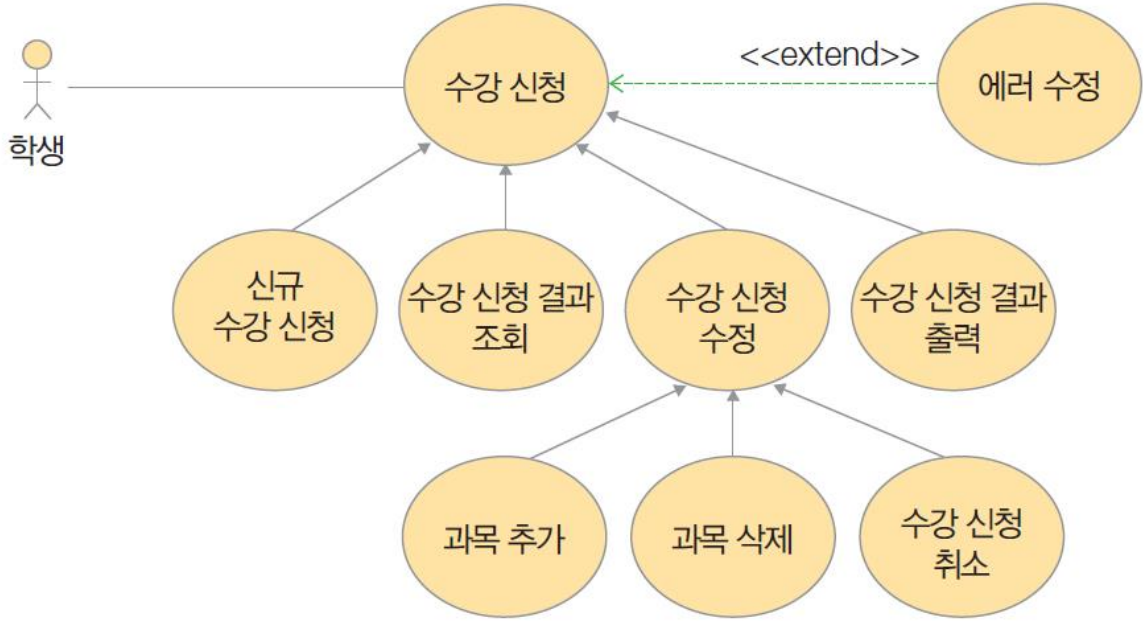
- 구조도 작성
  - 각 소프트웨어 모듈은 사각형, 모듈의 호출 관계는 화살표로 표시하여 계층적 트리 모양으로 작성

구조적 개발 방법

- 프로그램 설계 : 프로그램 모듈별로 알고리즘을 설계하는 것
- 자료 설계 : 자료를 저장할 파일이나 데이터베이스 구조를 설계하는 것
- 사용자 인터페이스 설계 : 사용자가 입력장치를 통해 시스템에 명령을 전달하고 시스템으로부터 응답 받는 방법을 설계하는 것
- 구현
  - 설계 단계에서 작성한 내용을 바탕으로 소스 코드를 작성하는 단계
- 테스트
  - 단위 테스트
  - 통합 테스트
  - 시스템 테스트
  - 인수 테스트

객체 지향 개발 방법

- 객체 지향 분석
  - 사용자의 요구를 유스케이스 다이어그램으로 작성하는 요구 추출과 추출된 요구를 분석하여 클래스 다이어그램과 순서 다이어그램으로 작성하는 요구 분석으로 나눠서 작업
- 요구 추출
  - 액터 찾기, 시나리오 작성, 유스케이스 작성 과정을 거침

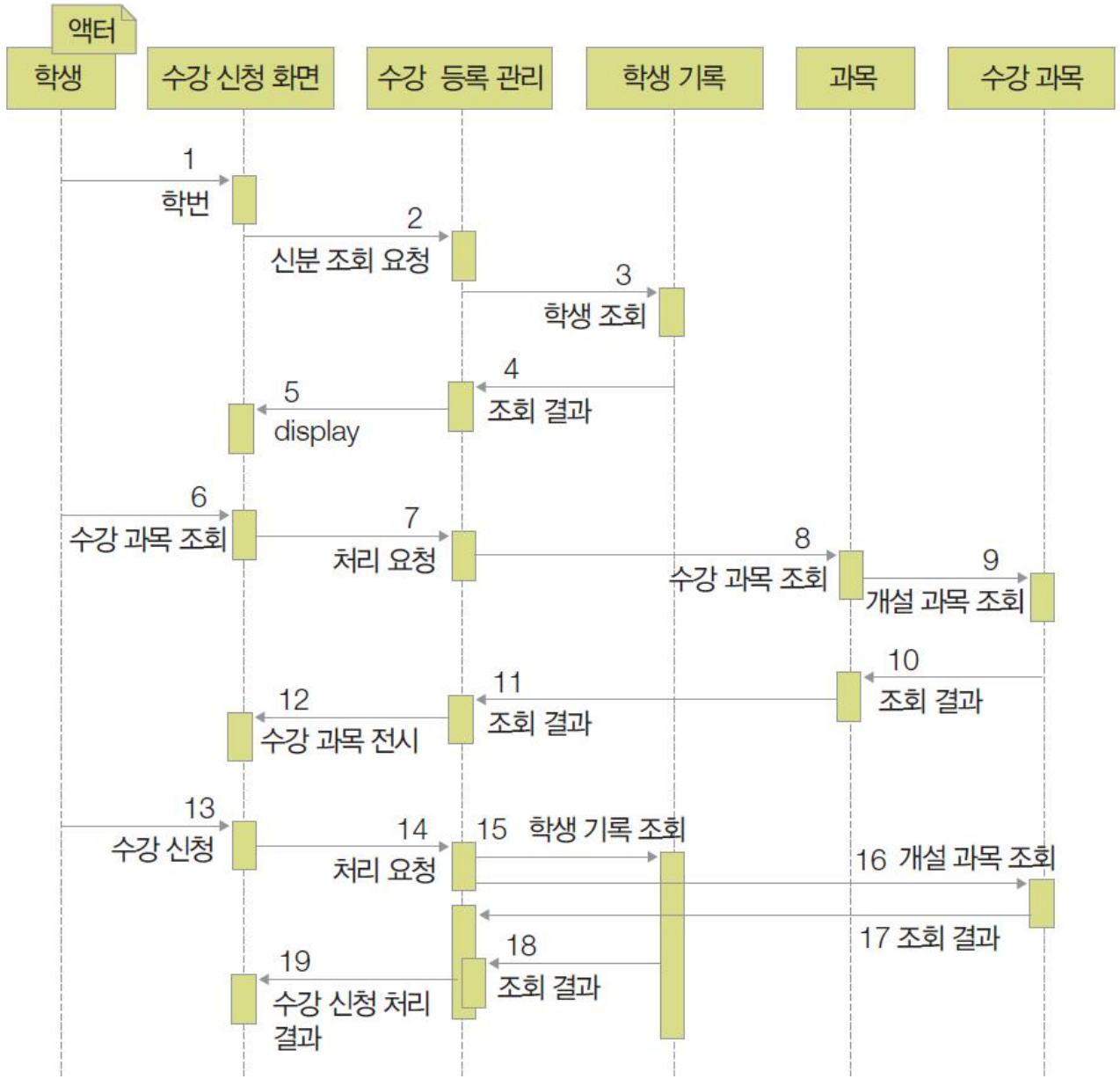


객체 지향 개발 방법

- 요구 분석
  - 클래스/객체 찾기
  - 클래스/객체 사이의 상호 작용 모형화
  - 클래스/객체 사이의 연관 관계 찾기
  - 객체 속성 추가

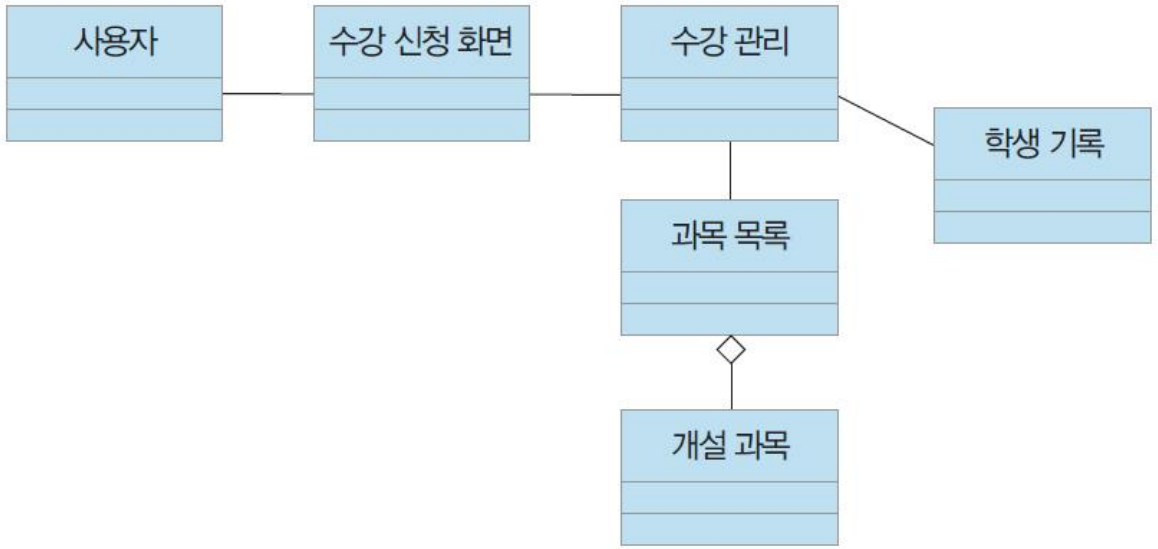
객체 지향 개발 방법

- 요구 분석



객체 지향 개발 방법

- 요구 분석



객체 지향 개발 방법

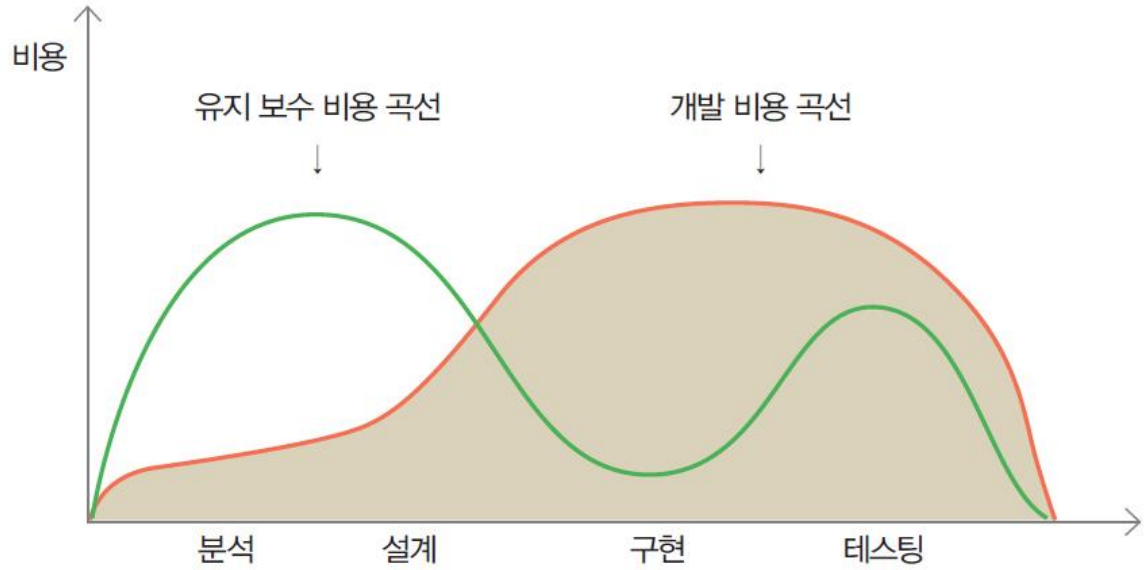
- 객체 지향 설계
  - 설정한 객체를 구체화하는 작업
  - 객체에 속한 속성이나 기능을 정의
  - 각 객체 사이의 메시지 전달 과정에서 빠진 객체와 기능을 보완해서 재정의

객체 지향 개발 방법

- 구현
  - 설계한 클래스/객체 상호 간의 메시지 전달 과정을 실제로 코딩하는 단계
  - 효과적인 구현을 위해 디자인 패턴과 리팩토링 방법을 사용
- 테스트
  - 메소드 테스트
  - 클래스/객체 테스트
  - 객체 집합 테스트
  - 시스템 테스트
  - +
  - 웹 특성을 고려한 추가 테스트

소프트웨어 유지 보수

- 유지 보수 절차
  - 분석
  - 설계
  - 구현
  - 테스팅
- 유지 보수 비용



소프트웨어 품질관리

- 소프트웨어 제품 품질
  - 기능성
  - 신뢰성
  - 사용 용이성
  - 효율성
  - 유지 보수성
  - 이식성
- 소프트웨어 프로세스 품질
  - SPICE
  - CMM

단계	수준	설명
5	최적화(optimizing)	현재는 물론 미래 프로세스에 적용해도 목표를 만족시키며 지속적으로 개선할 수 있는 수준
4	예측(predictable)	표준 프로세스가 일관된 통제에 따라 수행되며 정량적으로 성과를 측정할 수 있는 수준
3	확립(established)	표준으로 정해진 프로세스에 따라 결과가 산출되는 수준
2	관리(managed)	정해진 절차 및 계획에 따라 결과가 산출되는 수준
1	비정형적 실행(performed)	계획에 따라 꾸준히 결과물이 산출되는 것이 아니라 일시적으로 산출되는 수준
0	불안정(incomplete)	작업 산출물이나 결과를 기대하기 어려운 수준

웹 엔지니어링

- 웹 기반 소프트웨어와 일반 소프트웨어와의 차이
  - 네트워크 필수
  - 콘텐츠 중심
  - 지속적 진화

관점 지향 프로그래밍

- 절차 지향 프로그래밍에서는 분해된 관심을 프로시저로 구성했고, 객체 지향 프로그래밍에서는 클래스로 구성
- 객체 지향 프로그래밍에서 충분히 분리해낼 수 없는 관심이 존재한다는 문제에서 출발

항목	관점 지향 프로그래밍	객체 지향 프로그래밍
목적	부문별 조립을 통한 생산성 제고	상속 및 재사용을 통한 효율성 제고
산출물	관점	클래스
장점	환경 독립성, 횡단 관심 공통 적용	객체 재활용
단점	구현하고 이해하는 데 오랜 시간 소요	객체별 독립성 유지가 어려움

컴포넌트 기반 소프트웨어 개발

- 소프트웨어도 하드웨어처럼 교체가 필요한 부분을 갈아 끼우면 바로 동작하는 모델
- 컴포넌트의 크기와 적용 분야에 따른 분류
  - 사용자 인터페이스 컴포넌트
  - 기술 기반 컴포넌트
  - 비즈니스 기반 컴포넌트
  - 비즈니스 컴포넌트
  - 애플리케이션 컴포넌트