1 구조적 설계

01 구조적 설계

1 소프트웨어 설계 개요

소프트웨어 설계

- 구현에 앞서 데이터 베이스, 유저 인터페이스, 처리로직을 미리 강구하여 프로그램 개발의 청사진이 제시될 수 있도록 준비하는 활동

분석과 차이

- 분석은 문제를 정의(What)하고자 원칙을 적용하였다면, 설계에서는 문제를 해결(How)하고자 원칙을 적용함

01 구조적 설계

1 소프트웨어 설계 개요

🔎 설계에도 적용되는 원칙

01 구조적 설계

2 소프트웨어 설계 원칙

추상화

- 자세한 구상에 앞서, 상위 레벨에서 소프트웨어의 설계결과를 먼저 생각해보고 요약하는 것

추상화의 종류

절차 추상화	시스템을 기능적인 모듈로 나눔
데이터 추상화	데이터를 대표할 수 있는 표현으로 대체
제어 추상화	조건에 따라 다양한 경우의 수를 표현하는 것

01 구조적 설계

2 소프트웨어 설계 원칙

단계적 분해

- 문제를 해결하기 위해 “분할과 정복”이라는 개념을 적용하는 것
- 단계적 분해는 문제를 상위 개념부터 더 구체적인 단계로 하향식 분할하는 기법을 적용함

🔎 단계적 분해 과정

- 문제를 하위 수준의 독립된 단위로 나눔
- 구분된 문제의 자세한 내용은 가능한 뒤로 미룸
- 점증적으로 구체화 작업을 계속

01 구조적 설계

2 소프트웨어 설계 원칙

모듈화

- 수행 가능한 명령어를 잘라서 작은 독립단위로 나누어서 설계하는 것

모듈의 특징

- 자신의 이름을 가짐
- 독립적으로 컴파일됨
- 다른 모듈을 호출

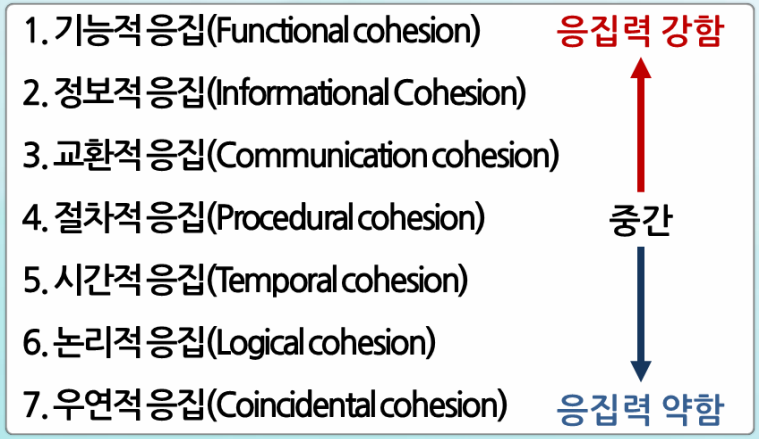
01 구조적 설계

2 소프트웨어 설계 원칙

모듈화

- 응집도
  - 모듈 내부에 수행하는 기능 간의 서로 관련되어 있는 정도

- 응집도의 특징
  - 강할 수록 좋은 설계
  - 응집도가 높으면 모듈 내부의 기능이 공통을 목적을 달성하기 위하여 관련성이 높음



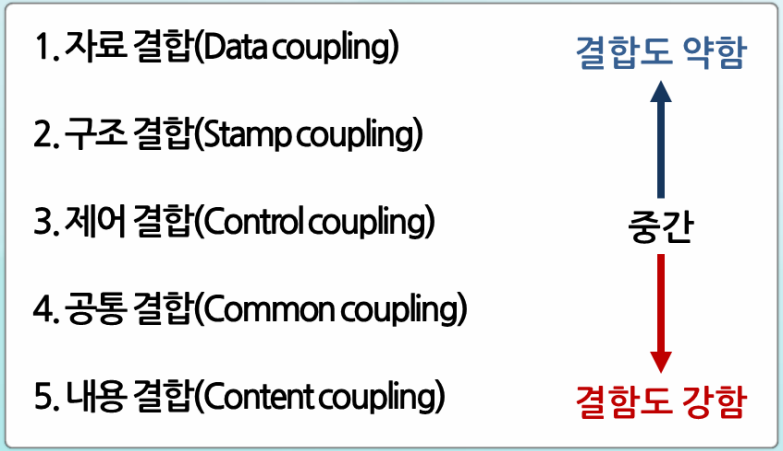
01 구조적 설계

2 소프트웨어 설계 원칙

모듈화

- 결합도
  - 모듈간에 연결되어 있는 상호 의존하는 정도

- 결합도의 특징
  - 낮은 결합도가 더 좋은 설계
  - 모듈끼리 서로 독립성을 갖도록 하고 결합도를 약하게 하는 것이 더 다루기 쉬움



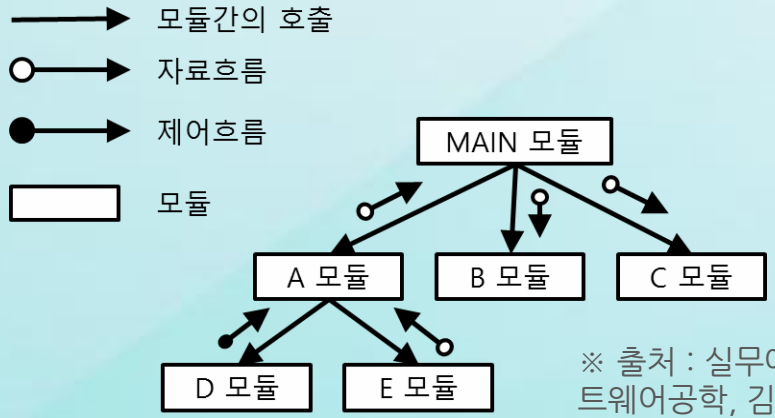
01 구조적 설계

3 구조적 설계 개요

- DFD를 기반으로 “구조도”를 작성하여 나타내는 것

※구조도란? : 소프트웨어를 어떤 모듈들로 나누었는지 보여주는 다이어그램

- 구조도에서 사용하는 기호
  - 모듈간의 호출
  - 자료흐름
  - 제어흐름
  - 모듈



※ 출처 : 실무에 바로 활용하는 소프트웨어공학, 김희영저, 21세기사

01 구조적 설계

4 구조적 설계 방법

변환 분석

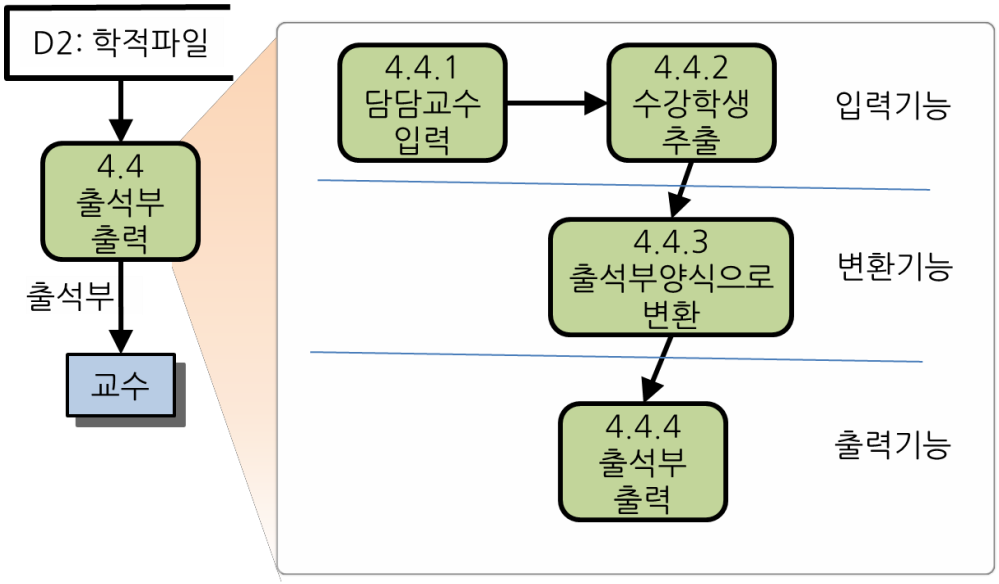
- 변환기능을 수행하는 모듈을 중심으로 입력모듈과 출력모듈을 찾아내는 분석과정
- 변환분석은 DFD를 입력기능, 변환기능, 출력기능으로 분할하여 시스템 구조도로 단계적인 계층을 만듦

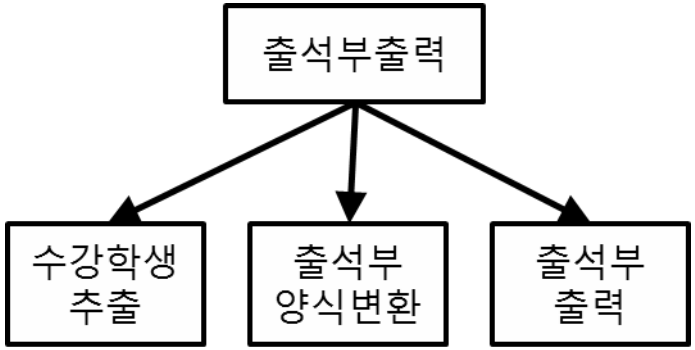
01 구조적 설계

4 구조적 설계 방법

변환 분석

※ 출처 : 실무에 바로 활용하는 소프트웨어공학, 김희영저, 21세기사





01 구조적 설계

4 구조적 설계 방법

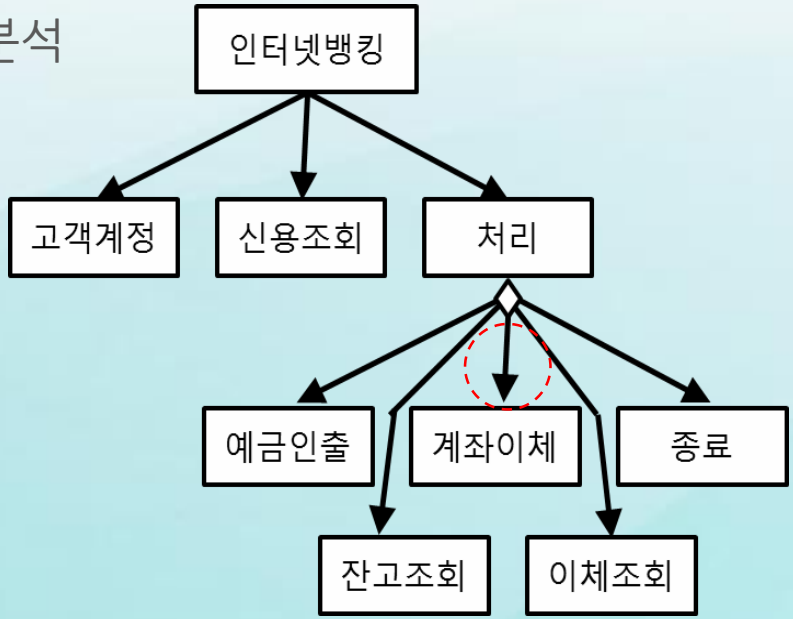
트랙잭션 분석

- DFD의 한 프로세스에서 여러 개의 데이터흐름이 유출되는 경우에 이를 분석하기 위해 활용
- 트랜잭션 분석은 새로운 기호인 “마름모꼴”을 추가함
- 마름모꼴의 의미는 여러 모듈 중에서 선택적인 호출을 할 수 있도록 하는 “선택”의 표시

01 구조적 설계

4 구조적 설계 방법

트랙잭션 분석



※ 출처 : 실무에 바로 활용하는 소프트웨어공학, 김희영저, 21세기사

2 객체지향적 설계

02 객체지향적 설계

1 객체지향 설계 개요

분석에서 설계로의 전환이
유기적 분석과 설계가 영속성을 가짐

객체지향 설계의 특징

- 분석에서 작성한 다이어그램을
  더욱 구체적이고 상세하게 나타내면 됨
- 클래스 추가, 메소드의 상세화,
  관계형 데이터베이스로 전환(OR매핑)

02 객체지향적 설계

2 유스케이스 설계

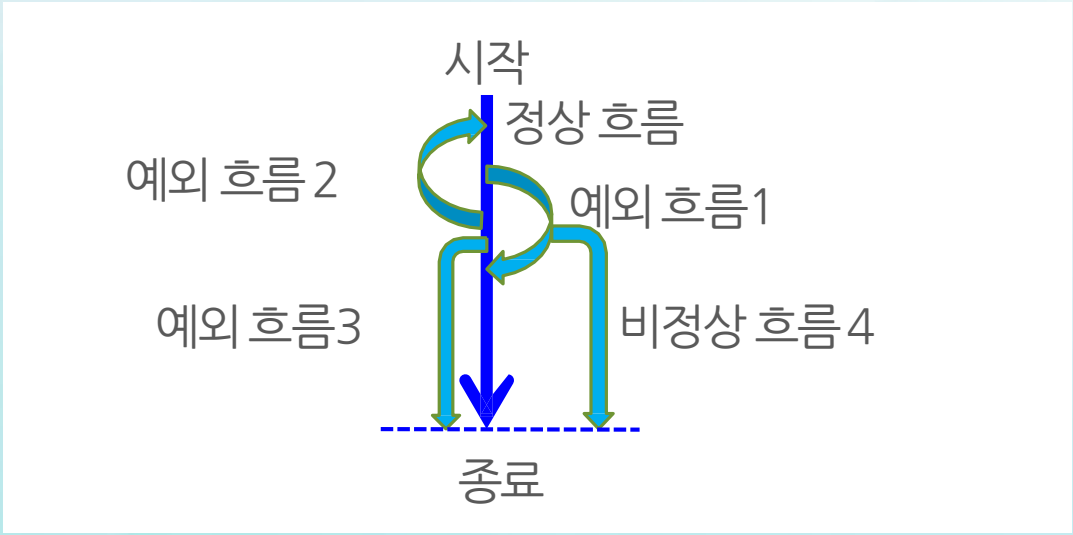
- 개별 유스케이스 단위 별로 유스케이스 분석서와 개발 표준에 의거하여 유스케이스 내부 시나리오를 설계
- 클래스간의 상호작용을 시퀀스 다이어그램으로 설계

02 객체지향적 설계

2 유스케이스 설계

유스케이스 내부 시나리오

- 정상 흐름 시나리오
- 예외 흐름 시나리오
- 비정상 흐름 시나리오



※ 출처 : 실무에 바로 활용하는 소프트웨어공학, 김희영저, 21세기사

02 객체지향적 설계

3 시퀀스 다이어그램 설계

- 구현 설계 요소들간의 상호 작용을 반영하여 다시 작성
- 구현이 가능한 수준으로 설계요소들에게 책임을 부여

02 객체지향적 설계

3 시퀀스 다이어그램 설계

🔎 설계 방법

- 공통적인 하부흐름은 없는지 체크
- 추상화 레벨을 맞추어야 하며, 새로운 하부흐름은 별개의 컴포넌트로 캡슐화
- 클래스, 속성, 메소드의 이름이 적절한지 체크
- 동일한 개념의 클래스는 합침
- 필요 시 추상화 클래스를 도출하여 상속관계 설정

02 객체지향적 설계

3 시퀀스 다이어그램 설계

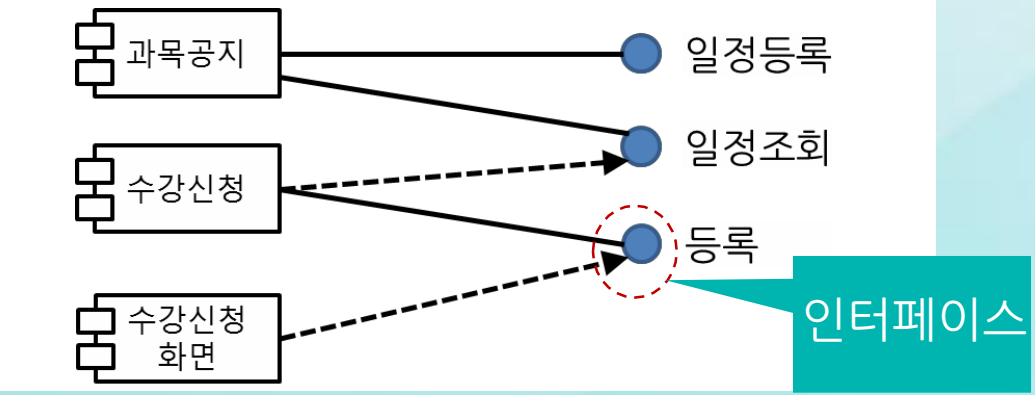
- 설계단계에서 추가적인 다이어그램이 추가될 수 있음
- 다이어그램의 추가
  - 컴포넌트 다이어그램
  - 디플로이먼트 다이어그램
  - 기타...

02 객체지향적 설계

4 컴포넌트 다이어그램

- 소프트웨어 컴포넌트 간의 종속관계, 혹은 모듈들 간의 종속관계를 보여줌

- 수강신청 일정을 생성하고 수강신청을 하는 절차를 나타내는 컴포넌트(예)

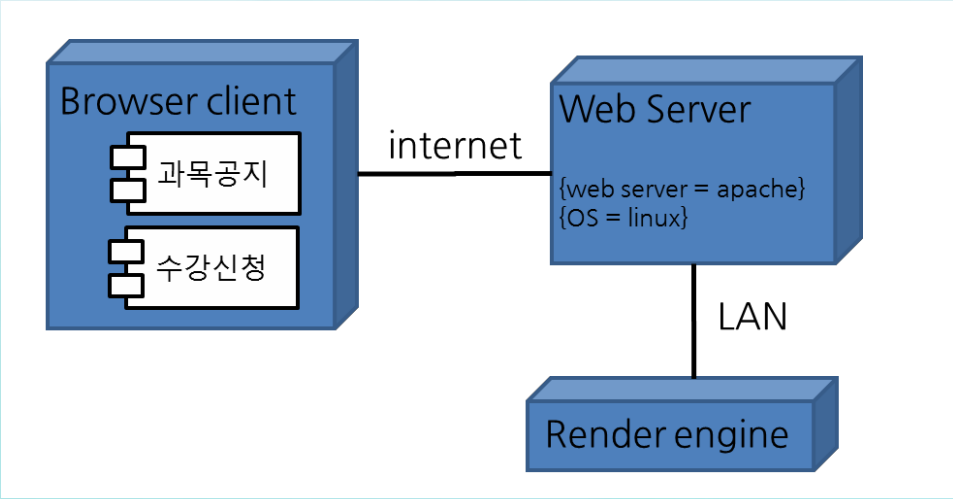


※ 출처 : 실무에 바로 활용하는 소프트웨어 공학, 김희영저, 21세기사

02 객체지향적 설계

5 디플로이먼트 다이어그램

- 시스템의 물리적인 아키텍처의 배치방식을 보여줌
- 디플로이먼트 다이어그램(예)

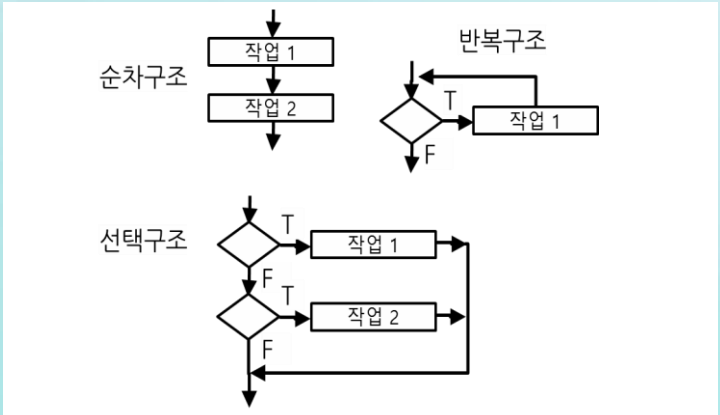


※ 출처 : 실무에 바로 활용하는 소프트웨어 공학, 김희영저, 21세기사

02 객체지향적 설계

6 모듈 내부의 설계

- 모듈 내부는 순차적이거나, 선택적이거나, 반복적이거나 → **알고리즘**
- 알고리즘에 의해 설계된 처리로직에 따라 데이터는 변환되거나, 계산되거나, 새로운 결과를 도출함



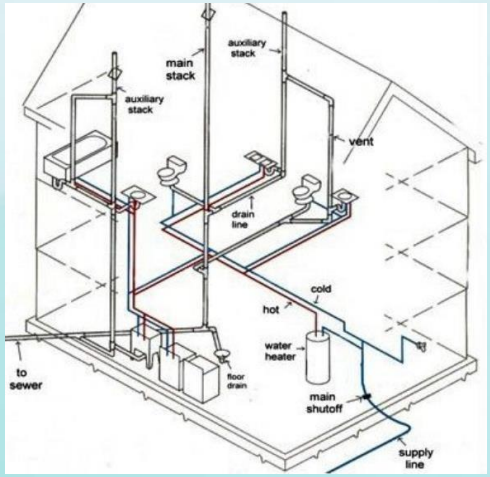
※ 출처 : 실무에 바로 활용하는 소프트웨어 공학, 김희영저, 21세기사

3 소프트웨어 아키텍처 설계

03 소프트웨어 아키텍처 설계

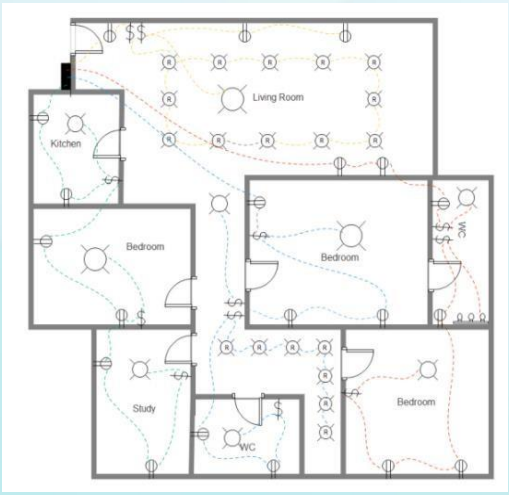
1 건축에서의 아키텍처

평면도만으로는 건물을 지을 수 없음 ➡ 다양한 도면이 필요



[상하수 배관도]

[평면도]



[배선도]

※ 출처 : 실무에 바로 활용하는 소프트웨어공학, 김희영저, 21세기사

03 소프트웨어 아키텍처 설계

2 소프트웨어 아키텍처와 모델링

- 시스템을 개발하기 위한 여러가지 관점의 제시가 필요

- 소프트웨어 공학에서의 아키텍처
  - 먼저, 전체 시스템의 전반적인 구조를 개략적으로 설계
  - 세부적인 서브시스템 혹은 소프트웨어 모듈들이 상호작용하는 방식을 이해하기 쉽게 분해

    ↳ 이렇게 보여진 결과를 “모델”이라고 함

03 소프트웨어 아키텍처 설계

2 소프트웨어 아키텍처와 모델링

소프트웨어와 시스템의 차이

- 많이 혼용하여 사용 함

소프트웨어	하드웨어에 대한 차별적인 용어
시스템	소프트웨어, 하드웨어, 네트워크 등 소프트웨어가 작동하도록 하는 환경 포함

03 소프트웨어 아키텍처 설계

3 소프트웨어 아키텍처 설계

- 시스템을 서브시스템으로 분해하여 서브시스템의 제어와 통신, 그리고 서브시스템끼리의 상호작용을 위한 프레임워크를 설정하는 설계 프로세스

- 소프트웨어 아키텍처 문서화의 장점
  - 의사소통
  - 시스템 분석
  - 재사용

03 소프트웨어 아키텍처 설계

3 소프트웨어 아키텍처 설계

🔎 소프트웨어 아키텍처를 설계하는 방식

구조적 방식 (필터 & 파이프)	기능지향적으로 시스템을 분해하는 방식
UML의 4+1 관점(View)	객체들로 분해하는 방식

03 소프트웨어 아키텍처 설계

4 소프트웨어 아키텍처의 4+1 관점

사용자, 분석가/설계자, 프로그래머, 시스템 통합자, 시스템엔지니어의 관점에 따라 아키텍처모델을 분리