강의노트

10주차 02차시

컴포넌트 기반 소프트웨어 개발

❖ 학습안내

이번 시간의 학습내용과 학습목표를 확인해보세요.

▣ 학습내용

- 개발방법론의 진화
- 컴포넌트 기반 소프트웨어 공학(CBSE)
- CBD개발 방법론

▣ 학습목표

- 개발방법론의 진화를 통하여 CBD를 설명할 수 있다.
- 컴포넌트 기반 소프트웨어 공학에 대하여 설명할 수 있다.
- 컴포넌트를 기반(CBD)으로 소프트웨어를 개발 할 수 있다.

❖ 학습내용

[1] 개발방법론의 진화

1. 개발 방법론

◈ 개발방법론의 개념

▪ 개발방법론의 정의

- 소프트웨어 공학원리를 소프트웨어 개발생명주기(SDLC)에 적용한 소프트웨어 개발방법
- 정보시스템을 개발하기 위한 작업활동, 절차, 산출물, 기법 등을 체계적으로 정리한 것

▪ 개발방법론의 필요성

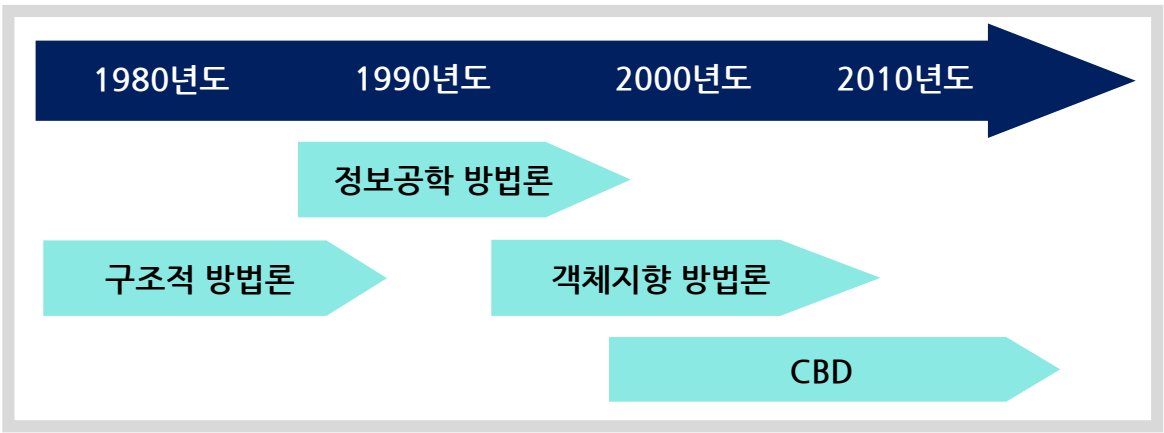
- 개발작업공정을 표준화 및 모듈화하여 개발경험 축적과 재사용을 가능하게 하여 개발 생산성 향상방안
- 수행공정을 관리 가능하게 가시화하여 효과적인 개발 및 관리방법을 제시
- 사용자 및 개발자간의 의사소통을 위한 수단으로 표준화된 용어가 필요

▪ 개발방법론의 시대별 변화

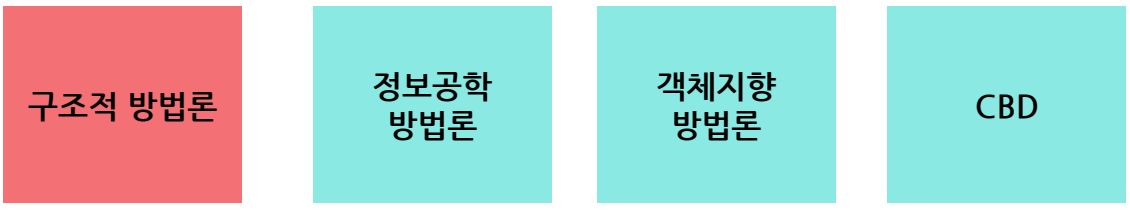
- 이전 개발방법론이 소멸된 것은 아니며, 현재 CBD방법론이 가장 활발히 활용됨

◈ 개발방법론의 시대별 변화

▪ 그림은 주 활용 개발 방법론에 대한 시대별 흐름



▪ 개발방법론의 특징



Logic중심, 제어 가능 모듈로 구조화 → 재사용 및 유지보수성 제고

❖ 학습내용

[1] 개발방법론의 진화

1. 개발 방법론(계속)

◈ 개발방법론의 개념(계속)

▪ 개발방법론의 특징

구조적 방법론	정보공학 방법론	객체지향 방법론	CBD

Logic중심, 제어 가능 모듈로 구조화 → 재사용 및 유지보수성 제고

구조적 방법론	정보공학 방법론	객체지향 방법론	CBD

고도의 모듈화, 상속에 의한 재사용(White Box Reuse)

구조적 방법론	정보공학 방법론	객체지향 방법론	CBD

Black Box Reuse 지향, 컴포넌트 생산/선택/평가/통합의 개발방법

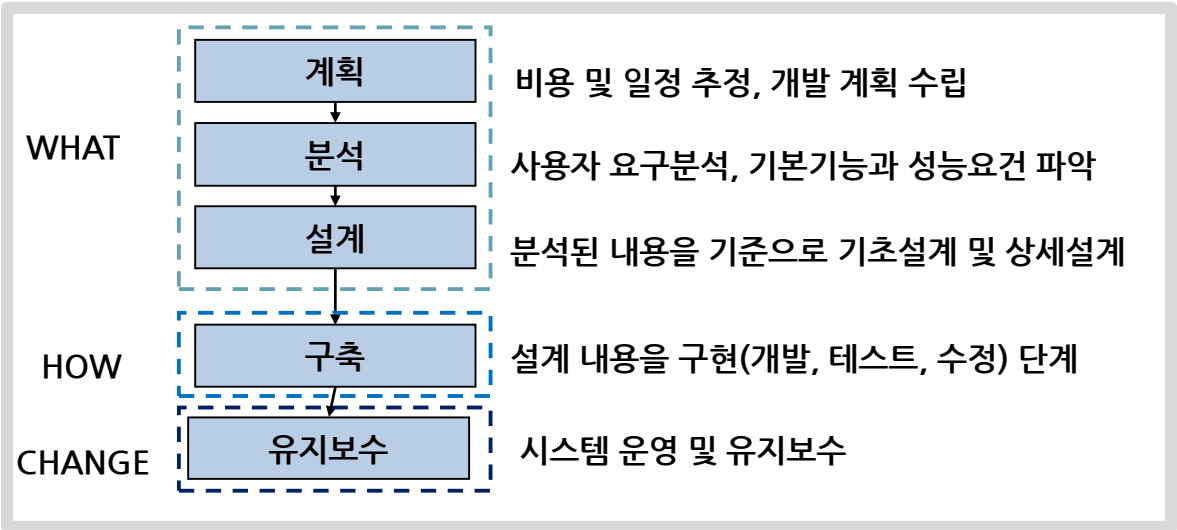
❖ 학습내용

[1] 개발방법론의 진화

1. 개발 방법론(계속)

◈ 개발방법론의 적용단계

▪ 소프트웨어 개발 단계별 사항들을 방법론에서 제시하고 있음



◈ 개발방법론의 구성 요소

▪ 방법론에는 다음 구성요소들을 제시하고 있음

구성요소	내 용	비 고
작업절차	▪ 프로젝트 수행 시 이루어지는 작업단계의 체계 ▪ 단계별 Activity의 정의, Activity별 세부작업 열거, Activity의 순서 명시	단계별 작업항목
작업방법	▪ 각 단계별 수행해야 하는 항목 정의 ▪ 절차와 작업방법을 명시(누가, 언제, 무엇을 작업하는지 기술)	작업방법
산출물	▪ 각 단계별로 만들어야 하는 산출물의 목록 및 양식	설계서 등

❖ 학습내용

[1] 개발방법론의 진화

1. 개발 방법론(계속)

◈ 개발방법론의 구성 요소(계속)

▪ 방법론에는 다음 구성요소들을 제시하고 있음

구성요소	내 용	비 고
관리	▪ 프로젝트의 진행 기록 ▪ 계획수립, 진행관리, 품질, 외주, 예산, 인력관리 등 기록	계획서, 실적, 품질보증 등
기법	▪ 각 단계별로 작업수행 시 기술 및 기법의 설명	구조적, 객체지향, ERD, DFD
도구	▪ 기법에서 제시된 각 기법 별 지원도구에 대한 구체적인 사용표준 및 방법	CASE 등

❖ 학습내용

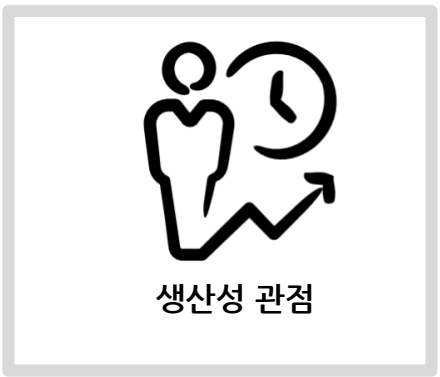
[1] 개발방법론의 진화

1. 개발 방법론(계속)

◈ 개발방법론 도입 시 고려사항

재사용 관점

- 수작업을 최소화하고 자동화되어 있을수록 좋음(시간과 비용)
- 프로젝트 결과물(모듈,설계서,컴포넌트) 재사용을 위한 정보공유체제 마련
- 의사소통, 형상관리, 품질관리를 위한 Repository 구축



생산성 관점

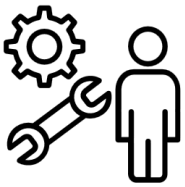
- 소프트웨어 개발 프로세스 능력향상 기대, 성능기대
- 검증된 결과물/컴포넌트를 사용하여 안정성 및 생산성 향상
- 프로젝트 특성(응용분야, 시스템 규모, 복잡도, 성격 등)을 고려하여 방법론을 선택해야 하고, 향후 유지보수 단계에서 사용 가능한 결과물을 선택하고 작성해야 함

❖ 학습내용

[1] 개발방법론의 진화

1. 개발 방법론(계속)

◈ 개발방법론 도입 시 고려사항(계속)



관리적 관점

- 최신 개발 방법론인 CBD 도입 시에는 인력을 ROLE기반으로 재배치 고려해야 함
- 소규모 프로젝트에 방대한 규모의 방법론 적용하는 것은 지양하고, 상황에 따라 우선 순위를 정해서 점진적 단계별로 추진필요
- 성공을 위한 가이드라인, 함정에 대한 경고 및 실제 활동에서 잊기 쉬운 점들을 체크 (통제수단과 산출물 인도방식)
- 개발자들에게 공감 하에 적절히 이용할 수 있어야 함(방법과 도구, 경험)

2. 구조적 방법론과 정보공학 방법론

◈ 구조적 개발 방법론

◈ 구조적 기법의 정의

- 업무활동 중심의 방법론으로 정형화된 절차 및 도형 중심의 도구를 이용하여 사용자 요구사항 파악 및 문서화하는 기법
- 구조적 방법론의 기본적인 뿌리는 구조적 프로그래밍에서 출발하여 설계의 원칙들을 정리한 구조적 설계, 시스템 복잡성 해결을 위한 구조적 분석으로 발전

◈ 특징

- 정보와 정보의 구조를 중심으로 분석, 설계, 구현
- 정형화된 분석절차에 따라 사용자 요구사항을 파악하고 도형중심의 다이어그램을 이용하여 문서화
- GOTO 분기 대신에 3개 논리적인 구조(Constructs)인 순차(Sequencing), 선택(Selection), 반복(Iteration)을 구성하여 프로그램 흐름 복잡성을 감소
- 생산성 향상, 품질 개선, 유지보수성 향상의 기여

❖ 학습내용

[1] 개발방법론의 진화

2. 구조적 방법론과 정보공학 방법론(계속)

◈ 정보공학 방법론

▪ 정보공학 방법론의 정의

- 기업 전체 또는 주요부문을 대상으로 정보시스템 계획수립, 분석, 설계, 구축에 정형화 된 기법들을 상호 연관성 있게 통합・적용하는 데이터 중심 방법론 기업에 필요한 정보와 업무를 총체적, 체계적, 효과적으로 파악하여 이를 모형화 하고 빠른 시간 내에 정보시스템으로 발전시키기 위해 필요한 일련의 작업 절차를 자동화한 공학적인 방법론

▪ 정보공학 방법론의 등장배경

환경의 변화

- 비즈니스의 변화: 컴퓨터 이용 활성화, 업무 기능 및 데이터의 분업화
- 정보기술의 발달: 하드웨어, 네트워크, R-DBMS 성능향상 등

▪ 정보공학 방법론의 등장배경

구조적 방법론의 한계 → 1990년대 초 James Martin이 제창

- 데이터 모델링 방법의 미흡, 기업 전반의 거시적 관점의 부족
- 명확한 방법론적 지침의 미흡, 설계와 코딩을 강조

▪ 정보공학 방법론의 특징

✓ 기업업무중심(ISP포함), 자료중심, 도형중심 접근

✓ 프로젝트 계획, 개발, 운영 단계의 명확한 구조기반 제시

✓ 정보시스템 개발의 자동화 지향

❖ 학습내용

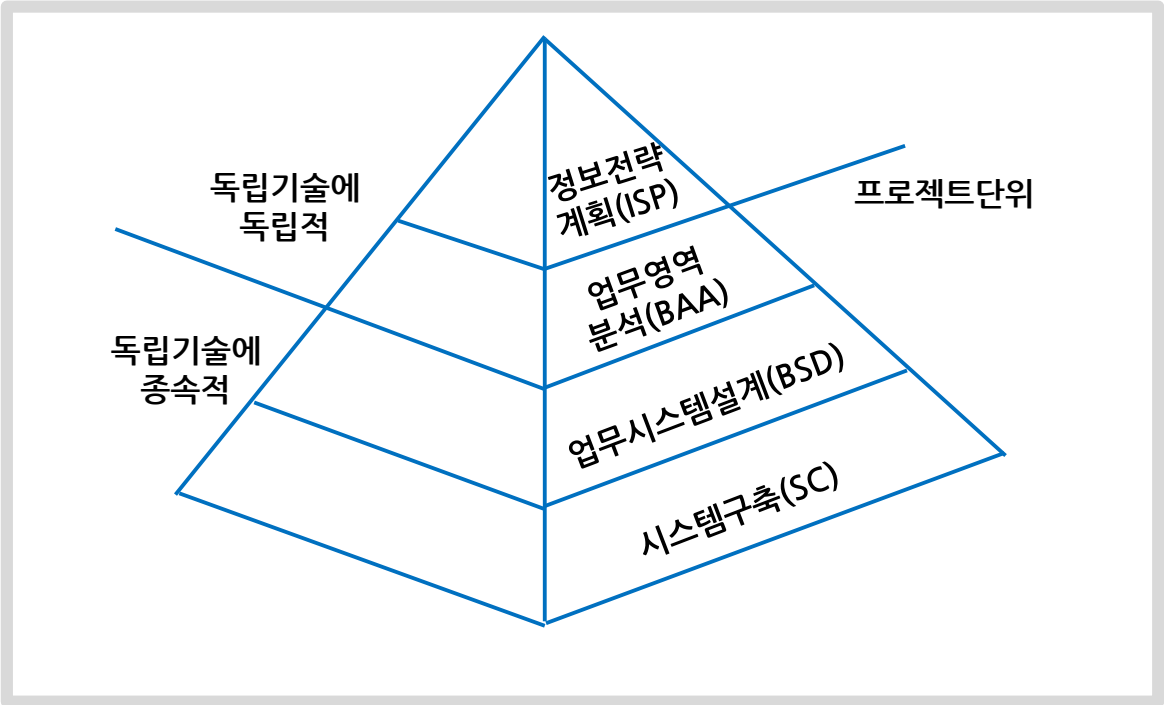
[1] 개발방법론의 진화

2. 구조적 방법론과 정보공학 방법론(계속)

◈ 정보공학 방법론(계속)

- 고객지향적, 최신 정보기술 능동적 수용
- 공학적 접근방식을 사용
- 적극적인 사용자 참여를 유도함

◈ 정보공학 피라미드



❖ 학습내용

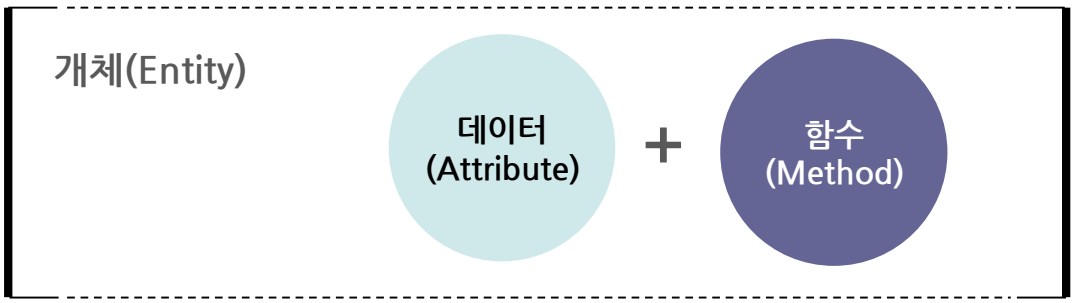
[1] 개발방법론의 진화

3. 객체지향 방법론과 CBD

◈ 객체지향 방법론의 개요

▪ 객체지향 방법론의 정의

- 현실세계에서 개체(Entity)를 데이터(Attribute)와 함수(Method)를 결합시킨 형태로 표현하는 개념으로 객체간의 메시지 통신을 통해 시스템을 구현하는 개발방법



◈ 객체지향 방법론의 기본개념

객체(Object)와 메시지(Message)

- 객체: 실 세계에 존재하는 사물을 표현하는 것으로 데이터와 함수로 구성
- 객체간의 통신은 메시지를 통하여 전달하며 외부객체에 의해 함수를 구현하여 객체의 데이터(Attribute)에 접근함

캡슐화(Encapsulation)와 정보은닉(Information Hiding)

- 객체의 데이터와 함수를 하나로 묶고 블랙박스화하여 외부와 접근을 제한함
- 정보은닉: 데이터의 임의변경을 통제하기 위해 메소드를 통해서만 접근이 가능토록 하는 것

❖ 학습내용

[1] 개발방법론의 진화

3. 객체지향 방법론과 CBD(계속)

◈ 객체지향 방법론의 기본개념 (계속)

클래스(Class)와 인스턴스(Instance)

- 클래스: 같은 종류 및 특성을 가진 객체들을 모아서 공통의 특성으로 분류하고 템플릿화하는 것

클래스(Class)와 인스턴스(Instance)

- 인스턴스: 클래스의 실체들로서 템플릿화된 클래스에서 파생된 하나의 실제 객체

예 붕어빵 틀 과 붕어빵을 생각하라.

상속(Inheritance)

- 상속: 클래스간의 IS-A 및 IS-PART-OF의 계층구조를 통하여 공통 특성을 상위 클래스로부터 물려받는 것
- 다중상속: 두 개 이상의 상위클래스로부터 상속으로 C++ 언어가 이를 지원
- 단일상속 : 오직 하나의 상위클래스로부터 상속가능하며 Java 언어지원

다형성(Polymorphism)

- 하나의 함수의 이름이나 연산자가 여러 목적으로 사용될 수 있는 것
- Overriding: 상위클래스에 정의된 Method를 하위 클래스에서 재정의
- Overloading: 매개변수의 데이터 형식에 따라 같은 이름의 Method 다중정의를 하여 여러 목적으로 사용함

❖ 학습내용

[1] 개발방법론의 진화

3. 객체지향 방법론과 CBD(계속)

◈ 객체지향 방법론의 개요

▪ 전통적 방법론과 객체지향 방법론의 비교

항목	구조적 개발 방법론	객체지향 개발 방법론
접근방법	Top Down	Bottom Up
설계방향	프로세스 중심(기능위주)	데이터중심(데이터+연산)
확장성 / 재사용성	확장 어려움 / 중복 많음	확장 용이 / 재사용성 높음
DBMS / CASE지원	전통적 DB(파일 및 관계형) / 상위레벨지원(다이어그램)	전통적 DB와 객체지향 DB 지원 / 상위레벨지원 (다이어그램)
방법 제시 모델	Jackson, Yourdon, Warnier-orr 제시한 모델 도구	UML(Booch, Rumbaugh - OMT, Jacobson - OOSE)

❖ 학습내용

[1] 개발방법론의 진화

3. 객체지향 방법론과 CBD(계속)

◈ CBD

▪ CBD의 정의

- CBD(Component Based Development)는 소프트웨어를 컴포넌트 개념으로 개발하고자 하는 방법론

▪ 소프트웨어 컴포넌트(Component)

- 공통 또는 특정 목적을 달성하기 위해 유사한 기능을 가진 어플리케이션들의 묶음
- 표준 인터페이스 정의를 가진 컴파일 된 이진형태의 코드
- 컴포넌트는 응용개발 시 선택 및 조립 통하여 개발 가능한 소프트웨어 조각모음
- 적당한 크기의 묶음을 통해 개발생산성 및 확장이 용이한 형태로 시장에 유통할 수 있게 표준 인터페이스를 원하는 소프트웨어

▪ 객체지향 방법과 CBD(Component Based Development)

항목	객체지향 프로그래밍	CBD
개발패턴	▪ 개발자들이 세부적으로 모든 프로그램을 개발하고 표준 부재 ▪ White Box 수준의 프레임워크	▪ 응용개발 시 개발자들은 미리 제공된 컴포넌트를 업무와 연관 지어 결합시키는 일에만 주안 ▪ 표준 인터페이스 제공
숙련도	▪ 개발자들은 객체지향 기술의 이해수준의 고급이어야 함 ▪ 모듈의 생산기술 수준	▪ 컴포넌트 전문가와 컴포넌트 조립 ▪ 응용개발자가 공동작업 가능 ▪ 개발자는 컴포넌트의 이해 및 조립
개발 프로세스	▪ 전통적 소프트웨어 개발 생명주기를 따름 ▪ 단계별 반복(Iteration) 없음	▪ 각 공정 별로 반복적인 프로세스가 있어 미니프로젝트가 가능함(Iteration)
상호 운용성	▪ 서로 다른 유형간의 상호 운영이 어려우며 특정 목적의 환경으로 개발될 가능성이 있음	▪ 다른 객체와 컴포넌트를 연결시켜 하나의 대형 객체를 생성가능 ▪ 표준화된 기술 적용

❖ 학습내용

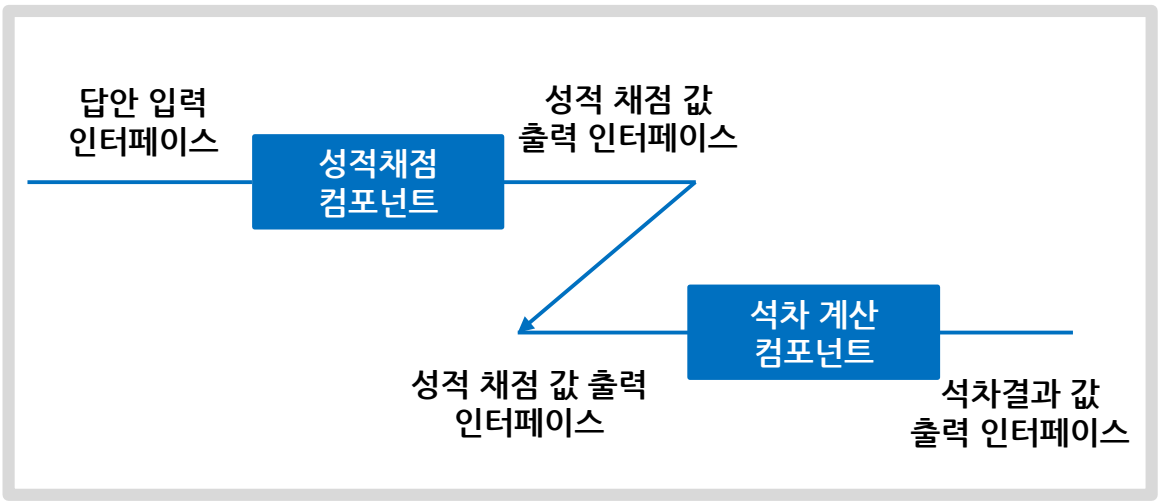
[2] 컴포넌트 기반 소프트웨어 공학

1. 컴포넌트와 컴포넌트 모델

◈ 소프트웨어 컴포넌트

▪ 소프트웨어 컴포넌트의 정의

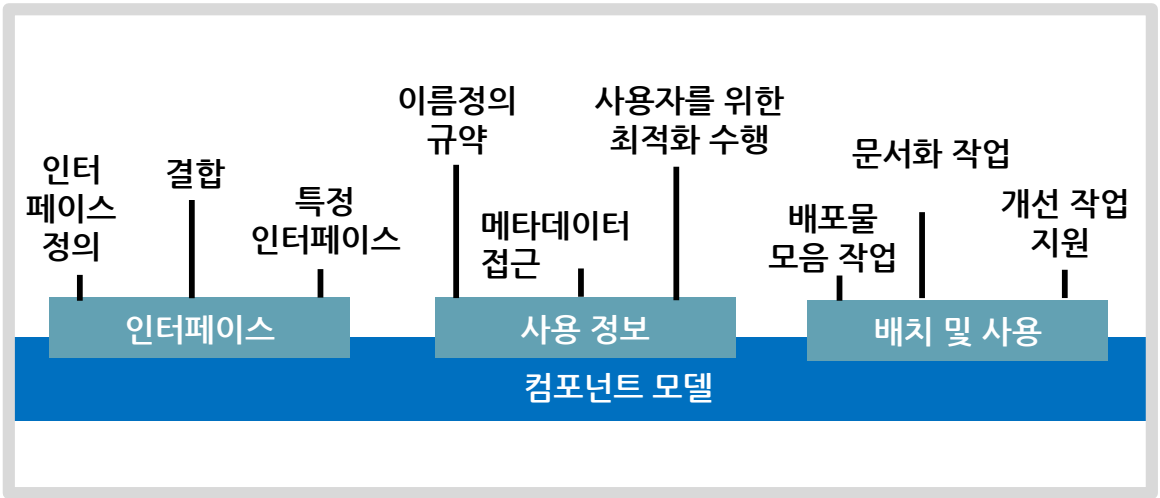
- 소프트웨어 시스템을 생성하기 위하여 다른 컴포넌트들과 결합될 수 있는 독립적인 소프트웨어 단위
- 컴포넌트는 다른 컴포넌트들과 결합하여 정보를 주고받는 인터페이스와 내부의 어떤 기능들을 처리하는 컴포넌트부분으로 나누어짐
- 컴포넌트 내부는 고치거나 내부를 자세히 볼 수 없는 블랙박스(Blackbox)형태로 관리



◈ 컴포넌트 모델

▪ 컴포넌트 모델 정의

- 컴포넌트의 구현, 문서화 배치를 위한 표준의 정의 등의 수행하는 방법을 정의한 것



❖ 학습내용

[2] 컴포넌트 기반 소프트웨어 공학

2. CBSE프로세스

◈ CBSE프로세스 정의

- 정의

  - CBSE: Component-based Software Engineering 컴포넌트 기반으로 소프트웨어를 개발하는 방법

- CBSE프로세스

  - 컴포넌트 기반 소프트웨어 공학(CBSE)의 프로세스는 컴포넌트 기반으로 소프트웨어를 개발하기 위하여 진행되는 절차를 의미

- CBD방법론

  - CBSE프로세스를 포함하며, 소프트웨어를 개발하기 위한 그 밖에 사항을 정의한 지침

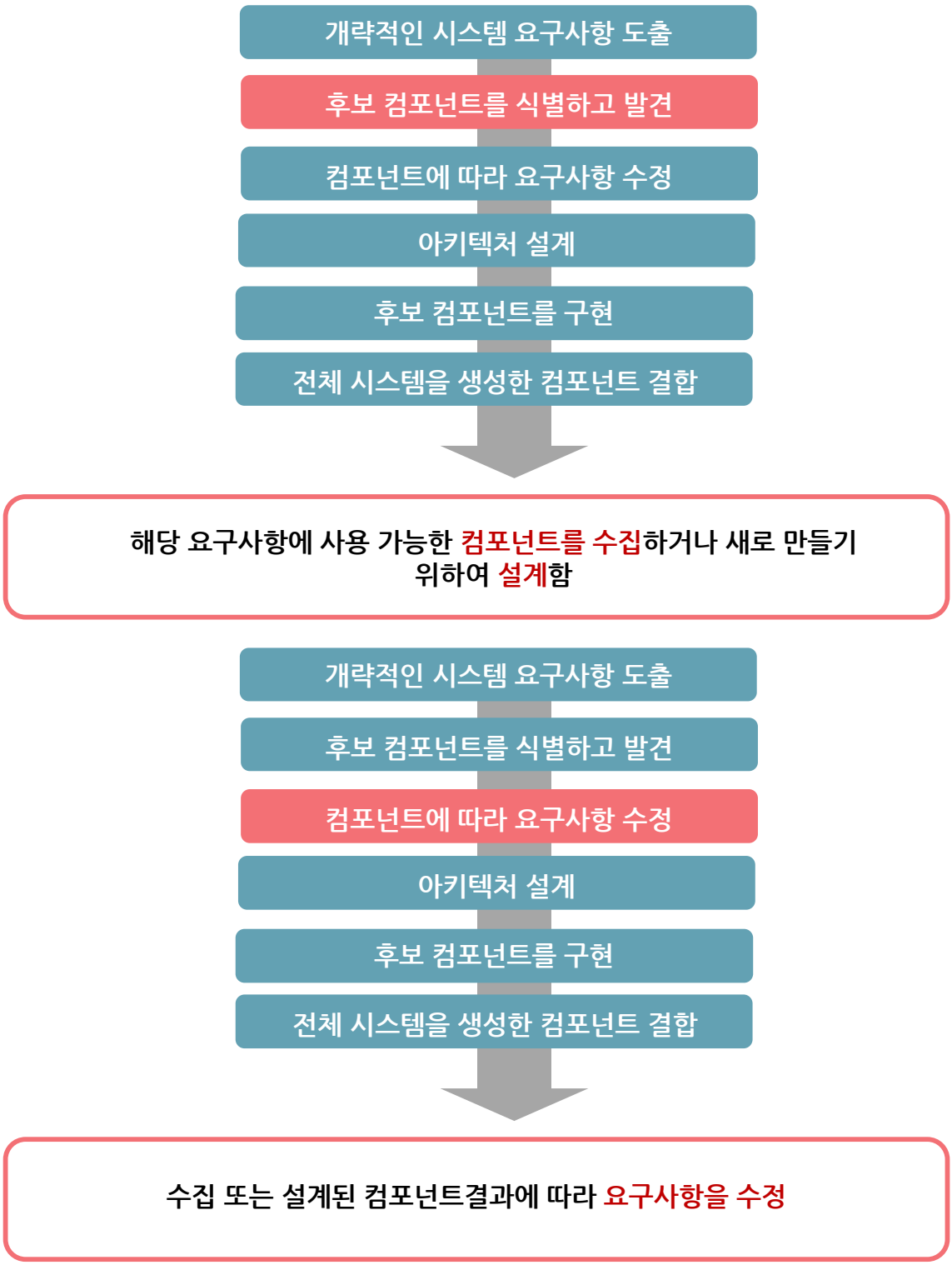
시스템 요구사항을 추후 변경을 감안하여 대략적으로 정의함

❖ 학습내용

[2] 컴포넌트 기반 소프트웨어 공학

2. CBSE프로세스(계속)

◈ CBD방법론(계속)

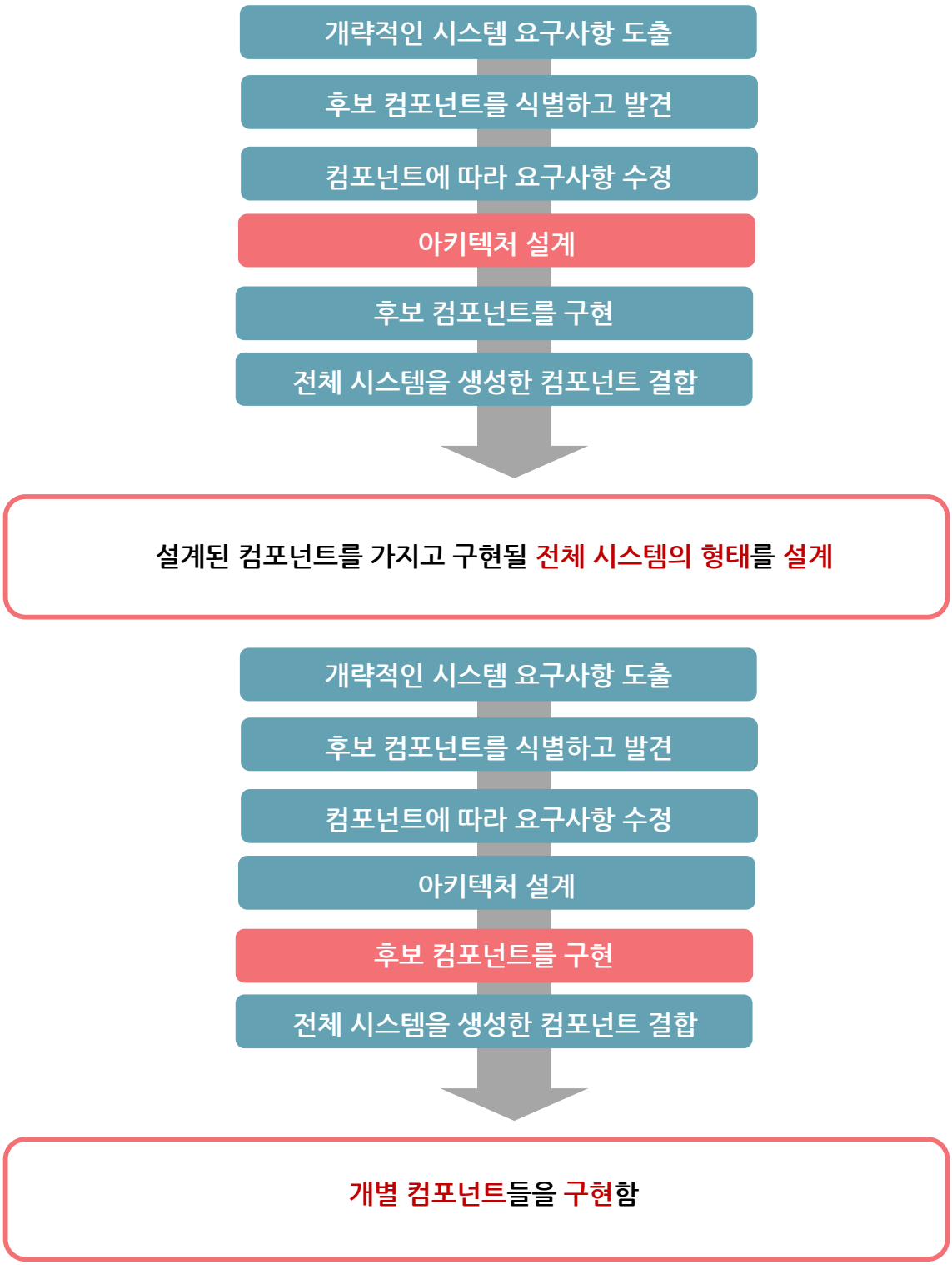


❖ 학습내용

[2] 컴포넌트 기반 소프트웨어 공학

2. CBSE프로세스(계속)

◈ CBD방법론(계속)

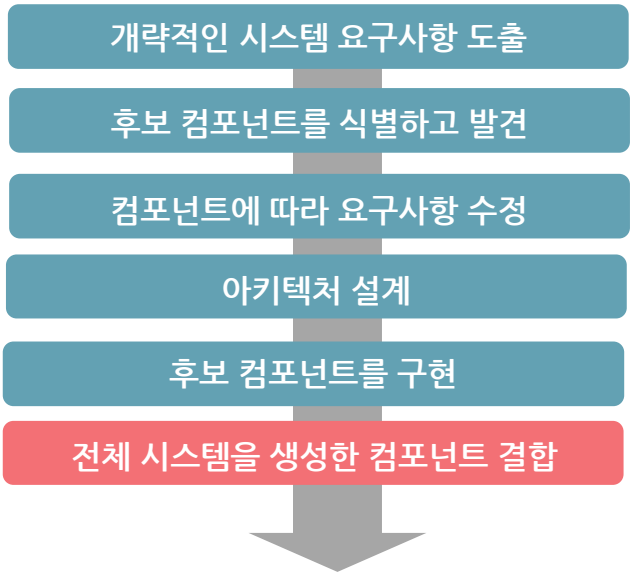


❖ 학습내용

[2] 컴포넌트 기반 소프트웨어 공학

2. CBSE프로세스(계속)

◈ CBD방법론(계속)



개별 컴포넌트들을 결합하여 전체 시스템을 완성함

3. 컴포넌트의 결합

◈ 컴포넌트 결합 방법

▪ 컴포넌트를 결합하여 전체 시스템을 완성하는 방법

순차적인 결합	계층적인 결합	부가적인 결합
순차적인 결합	계층적인 결합	부가적인 결합

- 구성컴포넌트들이 순차적으로 실행될 경우임
- 각 컴포넌트의 서비스 제공 인터페이스가 결합되며 컴포넌트 사이에 연결을 위하여 어떤 추가 코드가 필요

예 국어성적 채점 후 영어 채점 처리

❖ 학습내용

[2] 컴포넌트 기반 소프트웨어 공학

3. 컴포넌트의 결합(계속)

◈ 컴포넌트 결합 방법(계속)

▪ 컴포넌트를 결합하여 전체 시스템을 완성하는 방법(계속)

순차적인 결합 | 계층적인 결합 | 부가적인 결합

- 하나의 컴포넌트가 다른 컴포넌트가 제공하는 서비스를 직접 호출 시 발생
- 한 컴포넌트의 서비스 제공 인터페이스가 다른 컴포넌트의 서비스 요구 인터페이스와 결합되는 상황에 해당

예 국어 채점 후 국어점수 반 평균 처리

순차적인 결합 | 계층적인 결합 | 부가적인 결합

- 새로운 컴포넌트를 생성하기 위하여 둘 이상의 컴포넌트가 인터페이스를 종합할 때 발생
- 복합 컴포넌트의 인터페이스는 구성 컴포넌트의 모든 인터페이스를 종합하고 만일 필요하면 중복된 오퍼레이션을 제거하고 생성

예 국어 채점, 영어 채점, 수학 채점 후 각 학생의 석차를 처리

❖ 학습내용

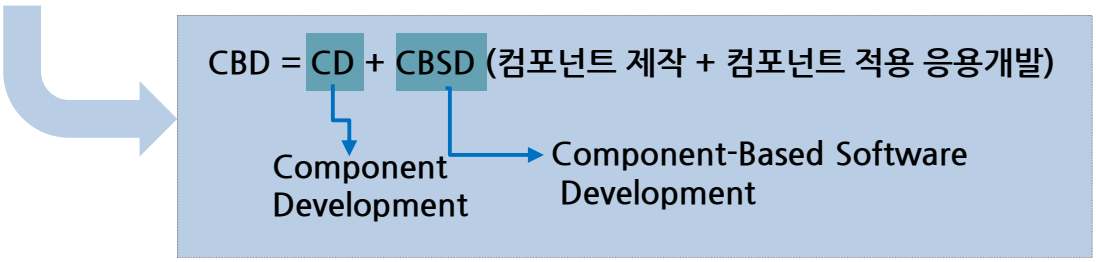
[3] CBD개발 방법론

1. 컴포넌트와 컴포넌트 모델

◈ CBD 방법론 개요

▪ CBD 방법론 정의

- 테스트가 완료된 소프트웨어 컴포넌트를 조립하여 사용자의 요구에 맞는 응용 소프트웨어를 만드는 방법으로 전통적 개발방법론 개념을 수용하면서 새로운(웹 기반 등) 개방형 아키텍처를 수용하려는 소프트웨어 공학적인 접근 개발방법
- 기존 객체지향 분석/설계에서 상속을 제외하고 인터페이스 중심의 접근을 강화한 재사용 프레임웍을 수용하는 방법론 임

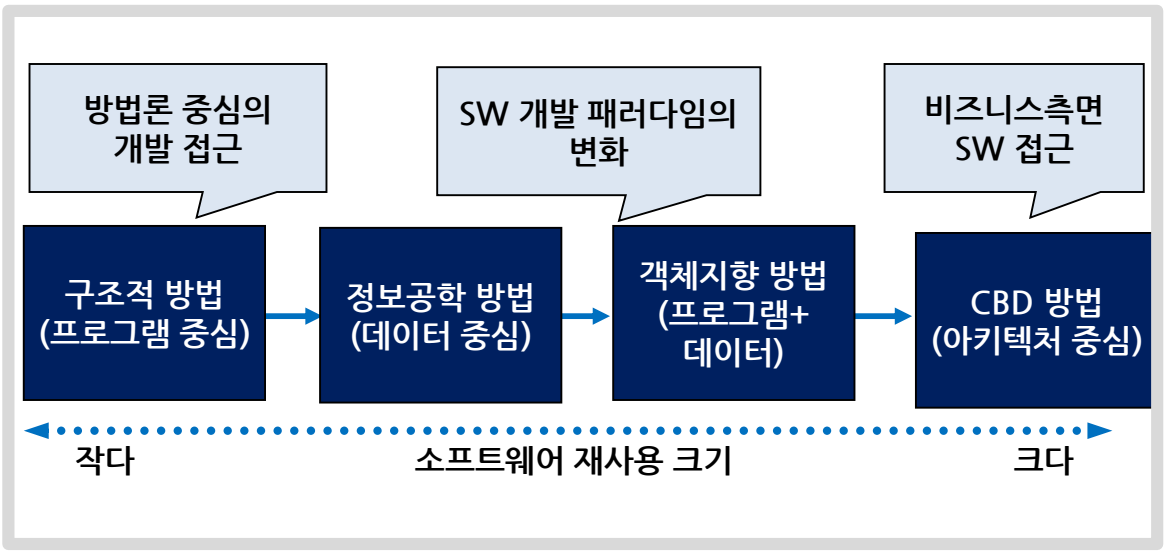


◈ CBD 방법론의 필요성

▪ 비즈니스 라이프사이클 타임에 적절히 대응할 필요가 있음(Time to Market)

▪ 빠르게 변화하는 비즈니스 환경에 능동적으로 대처(Flexibility)

▪ 네트워킹 및 통합을 위해 개방형 표준에 따른 정보시스템간 상호 운용성 필요



❖ 학습내용

[3] CBD개발 방법론

2. CBD 방법론 특징

◈ 특징

1 쓰임새주도(Use Case Driven)

- 프로젝트 이해당사자간 원활한 의사소통을 위해 UML(Unified Modeling Language)을 적용하며, 비즈니스 영역별로 현실에 맞게 쓰임새중심의 분석 및 설계단계 지원

2 아키텍처 중심(Architecture Centric)

- 소프트웨어의 재사용 및 개발의 생산성을 위해 프로젝트 시작과 함께 목적에 맞게 체계적인 아키텍처 계획을 수립하고 표준화 및 지속적인 개선노력을 병행함

선택(Select)	적합화(Adapt)	마무리, 개선 (Finalize & Evolution)

- 소프트웨어의 가시성(Visibility), 적응성(Adaptability)을 위해 컴포넌트 중심으로 웹기반 다 계층 아키텍처 등 다양한 환경에 적응함

3 반복과 점진(Iteration & Increment)

- 프로젝트 위험을 감소하기 위해 반복 계획 수립 시 목적을 명확히 하여 위험을 도출하며 계획대로 실행되었는지를 사용자 참여 하에 평가가 이루어짐

❖ 학습내용

[3] CBD개발 방법론

2. CBD 방법론 특징(계속)

◈ CBD 방법론과 객체지향 방법론

항목	CBD 방법론	객체지향 방법론
개발프로세스	소규모 단위의 프로젝트로 나누어 반복과 점진 수행 (Iteration & Increment)	전통적 SDLC를 따르며 개발 품질향상 을 위해 Prototype 수행 가능
아키텍처 측면	프로젝트 시작과 함께 계획 및 표준화 수립 후 지속적 개선	명확한 아키텍처 제시 및 표준화가 미흡함
응용개발 기술	컴포넌트 단위의 블랙박스 상태에서 표준 인터페이스 적용 (객체지향의 상속개념 없음)	객체지향 언어 적용 중심 프로그래밍(클래스 수준의 상속, 다형성 접근)
SW 공학 측면	비즈니스 중점의 소프트웨어 재사용을 통한 생산성 향상	데이터 및 프로세스 융합을 통한 개발 패러다임 변화

❖ 학습내용

[3] CBD개발 방법론

3. CBD 방법론 적용

◈ CBD 도입 전략

- 가능하면 많은 기능성의 서비스를 외부로부터 공급 받도록 하고 시험 완료된 검증된 컴포넌트로부터 시스템을 구성하고 기존의 설계기법들을 최대한 활용하는 비즈니스 측면에서 재사용 전략 수립
- 기업 개발조직의 성숙도 평가에 따른 CBD 수준 조정 및 목적공유를 통한 품질정책
- 표준화 기술을 통해 하나의 컴포넌트가 다양한 환경에서 활용될 수 있도록 상호운용성에 대한 고려

◈ CBD 도입 고려사항

프로젝트 관리측면

- 반복계획 수립은 명확한 목적과 평가가 이루어지도록 체크리스트를 작성하고 반복횟수는 프로젝트 상황을 고려하여야 함(많은 반복계획은 가능한 제한할 것)
- UML 다이어그램 적용을 통한 문서들의 가독성 향상방안 노력과 함께, 아키텍처는 표준 또는 유사한 프로젝트 플랫폼 모델을 참조하여 프로젝트 환경에 최적화 해야 함
- 조직의 체계화 전략과 비전을 수립하고 단계별 프로젝트관리 목표를 가지고 진행

개발 기술측면

- 공통계층의 컴포넌트 식별은 공통 유스케이스, 배타적 공통 클래스에 대해 클러스터링 과정 반복 수행
- 객체지향 설계에서 응용 Logic 재사용을 위한 컴포넌트 설계는 표현계층, 응용계층, 데이터계층이 일관성 있게 구성할 것을 권고

❖ 핵심정리

1. 개발방법론의 진화

- 개발방법론은 정보시스템을 개발하기 위한 작업활동, 절차, 산출물, 기법 등을 체계적으로 정리한 것
- 구조적 개발 방법론 → 정보공학 방법론 → 객체지향 방법론 → 컴포넌트 기반 개발 방법론 순으로 개발방법론이 등장하고 진화함

2. 컴포넌트 기반 소프트웨어 공학

- 컴포넌트 기반 소프트웨어 공학(CBSE)의 프로세스는 컴포넌트 기반으로 소프트웨어를 개발하기 위하여 진행되는 절차를 의미
- 컴포넌트를 결합하여 전체 시스템을 완성하는 방법은 순차적인 결합, 계층적인 결합, 부가적인 결합이 있음

3. CBD개발 방법론

- CBD방법론은 컴포넌트제작과 컴포넌트 적용 응용개발로 진행
- CBD방법론은 쓰임새주도, 아키텍처 중심, 반복과 점진의 특징을 가짐
- CBD방법론을 활용하기 위하여 가능하면 많은 기능성의 서비스를 외부로부터 공급받도록하고 시험 완료된 검증된 컴포넌트로부터 시스템을 구성하고 기존의 설계기법들을 최대한 활용하여야 함