애플리케이션 설계 방법론과 UML

한국기술교육대학교
온라인평생교육원

학습내용

- 애플리케이션 설계 방법론의 개요
- 애플리케이션 설계 방법론의 유형
- UML의 개요
- UML Diagram의 유형

학습목표

- 애플리케이션 설계 방법론의 개념을 설명할 수 있다.
- 애플리케이션 설계 방법론의 유형별 특징을 설명할 수 있다.
- 애플리케이션 설계에 필요한 UML의 개념을 설명할 수 있다.
- UML Diagram의 유형별 특징을 설명할 수 있다.

애플리케이션 설계 방법론의 개요

애플리케이션 설계 방법론의 개념

- 애플리케이션 개발에 대한 여러 작업 단계들의 수행방법(Method)
- 작업 수행 시 도움이 되는 기법(Technique) 및 도구(Tool)들을 이용
- 개발 경험을 바탕으로 각 작업 단계를 체계적으로 정리한 작업수행의 표준 규범

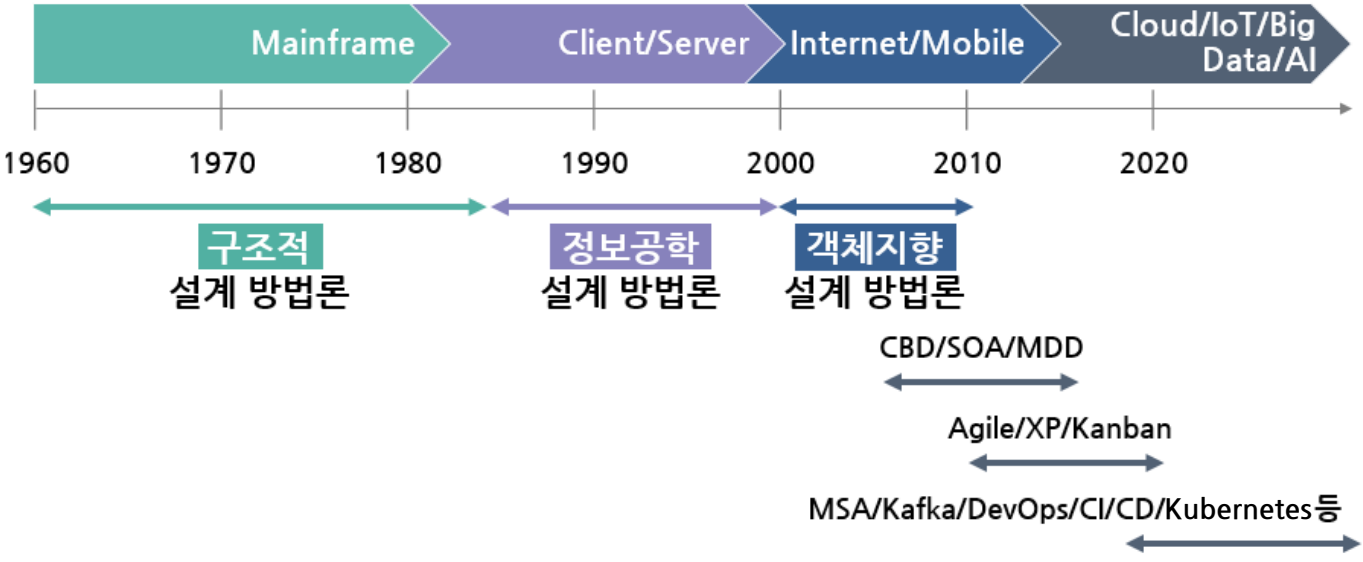
애플리케이션 설계 방법론의 개요

애플리케이션 설계 방법론의 구성요소

구성요소	내용	비고
작업절차 (Process)	▪ 프로젝트 수행 시 이루어지는 작업단계의 체계 ▪ 단계별 활동, 활동별 세부작업 열거, 활동의 순서 명시	단계-활동-작업
작업방법 (Method)	▪ 각 단계별 수행해야 하는 일에 대한 구체적인 설명 ▪ 절차·작업방법(누가, 언제, 무엇을 작업하는지를 기술)	작업방법
산출물 (Outputs)	▪ 각 단계별로 만들어야 하는 산출물의 목록 및 양식	설계서 등
기법 (Technique)	▪ 각 단계별로 작업 수행 시 소요되는 기술 및 기법 설명	객체지향 모델링 기법
도구 (Tools)	▪ 기법에서 제시된 각 기법별 지원도구에 대한 구체적인 사용표준 및 방법	CASE 등

애플리케이션 설계 방법론의 개요

애플리케이션 설계 방법론의 역사



애플리케이션 설계 방법론의 유형

- 구조적 설계 방법론
- 정보공학 설계 방법론
- 객체지향 설계 방법론
- 컴포넌트 기반 설계 방법론
- 모델 기반 설계 방법론

애플리케이션 설계 방법론의 유형

구조적 설계 방법론

정의

- 전체 시스템을 기능에 따라 분할·개발하고, 이를 통합하는 분할과 정복 접근 방식의 방법론

특징

- 데이터 플로우 다이어그램, 엔티티 관계도, 상태전이도, 미니스펙 등의 분석 도구 활용
- 프로세스 중심의 하향식 방법론
  - ➡ 최상위 구조에서 하향식으로 구체화하여 설계
- 애플리케이션 구조가 계층적으로 설계됨
  - ➡ Multi Level 데이터 플로우 다이어그램 활용

애플리케이션 설계 방법론의 유형

구조적 설계 방법론

단계별 구성요소 분석 → 설계 → 개발

기본원칙	분석도구	산출물
▪ 도형 중심 ▪ 분할과 정복 ▪ 정형화 ▪ 구조적 조직화 ▪ 하향식 기능분해	DFD(Data Flow Diagram)	자료흐름도
	ERD(Entity Relationship Diagram)	개체-관계도
	STD(State-Transition Diagram)	상태전이도
	Mini-Spec	소단위명세서
	DD(Data Dictionary)	자료사전

애플리케이션 설계 방법론의 유형

구조적 설계 방법론

단계별 구성요소 분석 → 설계 → 개발

기본원칙	분석도구	산출물
▪ 복합설계 기본 원칙(결합도, 응집도) ▪ SW 기능과 프로그램 구조/모듈 설계를 위한 전략, 평가지침, 문서화 도구를 지원하는 체계적 설계 기법	구조도 (Structure Chart)	N-S(Nassi-Schneiderman) Chart 등 로직 기술 도구
	프로그램 명세서 (절차 명세서)	
	Application 구조도	
	Multi Level DFD	Sub System, Functional Module, Program의 계층적 분류
	Database Table 기술서	

단계별 구성요소 분석 → 설계 → 개발

기본원칙
▪ 계층적 형식, 제한된 제어 구조, 작성순서대로 PGM 실행 ▪ 3개의 논리적 구조 : 연속(Sequence), 조건(IF-Then-Else), 반복(Repetition)

애플리케이션 설계 방법론의 유형

정보공학 설계 방법론

정의

- 기업 전체 또는 주요 부문을 대상으로 정보시스템 계획 수립, 분석, 설계, 구축에 정형화된 기법들을 상호 연관성 있게 통합·적용하는 데이터 중심 방법론

특징

- 기업 중심으로 정보전략 계획(ISP) 포함
- 데이터 중심의 분석과 설계 진행
- 도형 중심의 산출물
- 프로젝트를 관리 가능한 단위로 분할·정복
- 공학적 접근방식 사용
- 적극적인 사용자 참여 유도
- 정보시스템 개발의 자동화 지향

애플리케이션 설계 방법론의 유형

객체지향 설계 방법론

정의

- 애플리케이션 구조를 객체 기반으로 설계하는 방법
- 요구분석, 설계, 구현, 시험 등의 소프트웨어 생명주기에 객체지향 개념을 접목시켜 일관된 모델을 가지고 애플리케이션을 개발하는 방법론

특징

- 2000년대 초반 Java, C++같은 객체지향언어가 국내에 도입되기 시작하면서 사용됨
- 정보와 프로세스를 객체라는 하나의 대상에 통합하여 설계
- 객체를 기반으로 기능, 정적, 동적 모델링을 통하여 애플리케이션 설계

애플리케이션 설계 방법론의 유형

객체지향 설계 방법론

단계별 작업항목

단계	작업항목	설명
객체지향 분석	객체모델링 - 객체다이어그램	▪ 시스템 정적구조 포착 ▪ 추상화, 분류화, 일반화, 집단화
	동적모델링 - 상태다이어그램	▪ 시간 흐름에 따라 객체 사이의 변화 조사 ▪ 상태, 사건, 동작
	기능모델링 - 자료흐름도	▪ 입력에 대한 처리결과 확인
객체지향 설계	시스템 설계	▪ 시스템구조를 설계 ▪ 성능최적화 방안, 자원분배방안
	객체 설계	▪ 구체적 자료구조와 알고리즘 구현
객체지향 구현	객체지향언어 (객체, 클래스)로 프로그램	▪ 객체지향언어(C++, JAVA), 객체지향DBMS

*클래스 : 공통적인 특징을 갖는 객체를 모아 놓은 것

애플리케이션 설계 방법론의 유형

컴포넌트 기반 설계 방법론(CBD, Component Based Development)

정의

- 컴포넌트 단위의 개발·조립·유지보수
  - 정보시스템의 신속한 구축, 변경 확장의 용이성, 타 시스템과의 호환성을 달성

특징

- 2000년대 중반부터 국내에 활용되기 시작
- 컴포넌트를 구축하는 CD단계와 구축된 컴포넌트를 조립하는 CBD단계로 구분

CD(Component Development)

- 컴포넌트를 설계·구현한 후 이를 컴포넌트 저장소에 저장

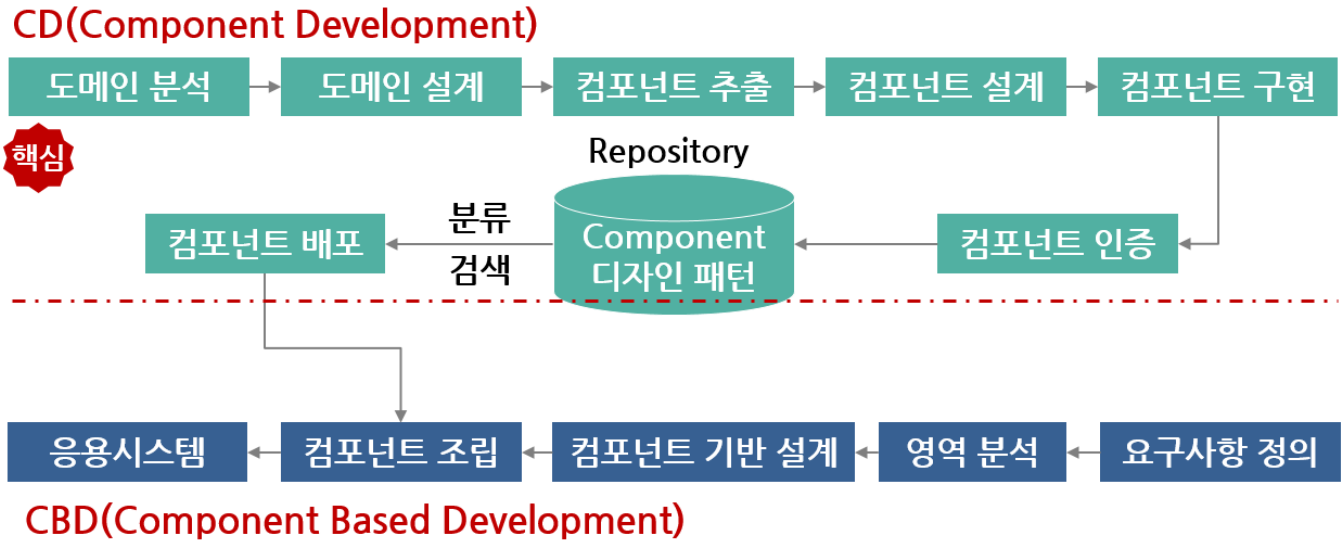
CBD(Component Based Development)

- 컴포넌트 저장소에서 필요한 컴포넌트를 추출한 뒤, 이를 조립하여 애플리케이션 구축

애플리케이션 설계 방법론의 유형

컴포넌트 기반 설계 방법론(CBD, Component Based Development)

개발 절차



애플리케이션 설계 방법론의 유형

모델 기반 설계 방법론(MDD, Model Driven Development)

정의

- 애플리케이션을 모델 기반으로 설계·구축하는 방법론
- 플랫폼 독립적인 아키텍처 모델로부터 플랫폼 종속적 애플리케이션 모델로 자동 변환하는 기술
  - MDA(Model Driven Architecture) 기반의 애플리케이션 설계 방식

애플리케이션 설계 방법론의 유형

모델 기반 설계 방법론(MDD, Model Driven Development)

CIM

과정	구현독립적 모델 (CIM, Computation Independent Model)
설명	▪ 비즈니스 분석 및 플랫폼 독립적인 분석 기능 구현
역할	▪ 업무분석가

PIM

과정	플랫폼 독립적 모델 (PIM, Platform Independent Model) 작성
설명	▪ 플랫폼 독립적인 모델 설계 ▪ 기본 서비스(Pervasive Service) 및 영역 특성에 대한 내용 포함
역할	▪ 아키텍트

애플리케이션 설계 방법론의 유형

모델 기반 설계 방법론(MDD, Model Driven Development)

PSM

과정	플랫폼 종속적 모델 (PSM, Platform Specific Model) 작성
설명	▪ 자동화 도구를 이용하여 PIM을 PSM으로 변환 ▪ PSM을 기술하기 위한 UML Profile(UML4EJB, UML4CORBA 등)에 부합되도록 매핑하여 변환
역할	▪ 시스템 아키텍트

Code

과정	Code 생성
설명	▪ 자동화 도구(Code Generator)를 이용하여 구현 환경에 적합한 코드 생성
역할	▪ Source Code 산출물

UML의 개요

UML의 개념

- 객체 기술에 대한 표준화 기구 (OMG, Object Management Group)에서 인정한 객체지향 분석, 설계를 위한 모델링 언어 (모델을 표현하는 언어)

UML의 특징

가시화 언어
- 그래픽 언어
- 의사소통 용이

명세화 언어
- 분석, 설계 내용 표현

구축 언어
- 다양한 언어와 연결

문서화 언어
- 의사소통 문서

UML의 개요

UML의 구성요소

사물(Things)

구조 사물

클래스(Class)	• 동일한 속성, 오퍼레이션, 관계, 의미를 공유하는 객체 표현
인터페이스(Interface)	• 외부적으로 가시화되는 요소의 행동 표현
객체(Object)	• 클래스의 인스턴스
유스케이스(Use Case)	• 시스템이 행위자에게 제공하는 기능 표현
액티브 클래스(Active Class)	• 쓰레드나 프로세스를 갖는 클래스를 표현
컴포넌트(Component)	• 물리적으로 관리되는 요소들의 패키지를 표현
노드(Node)	• 실행 시 나타나는 컴퓨터나 주변기기 등을 표현

행위 사물

상호작용(Interaction)	• 객체 사이의 메시지, 활동 순서 연결을 표현
상태 머신(State machine)	• 객체의 상태와 변화 순서 표현

그룹 사물

패키지(Package)	• 요소를 그룹으로 묶어 표현

주해 사물

노트(Note)	• 주석 및 제약을 표현

UML의 개요

UML의 구성요소

관계(Relationship)

연관관계(Association)	• 구조적 관점에서의 사물들의 연결 표현
포함관계(Aggregation)	• 사물 사이의 포함관계 표현
일반화관계(Generalization)	• 사물 간의 특수화, 일반화 관계
의존관계(Dependency)	• 사물 간의 영향을 미치는 관계
실체화관계(Realization)	• 사물의 수행하기로 되어 있는 계약을 명세화

UML의 개요

UML의 구성요소

다이어그램(Diagram)

기능모델

유스케이스 다이어그램	• 액터와 시스템을 사용하는 다양한 경우, 즉 유스케이스의 관계를 구조적으로 표현

정적모델

클래스 다이어그램	• 시스템을 구성하는 클래스, 인터페이스 사이의 구조적 연관관계를 표현
객체 다이어그램	• 특정 시점의 객체들의 구조적 상태를 표현
컴포넌트 다이어그램	• 소프트웨어의 물리적 단위의 구성과 의존 관계를 표현
배치 다이어그램	• 노드에 존재하는 컴포넌트의 물리적 구성을 표현

동적모델

순서(절차) 다이어그램	• 시스템 외부 이벤트를 처리하기 위해 시스템 내부 객체 간에 주고 받는 동적 메시지를 시간의 흐름에 따라 표현
협력 다이어그램	• 순서 다이어그램과 동일한 내용을 객체 상호 관계의 관점에서 표현
활동 다이어그램	• 시스템 내부의 활동 흐름을 표현
상태차트 다이어그램	• 시스템 내부의 상태 전이를 표현

UML Diagram의 유형

- Use Case Diagram
- Class Diagram
- Sequence Diagram
- Component Diagram

UML Diagram의 유형

Use Case Diagram

정의 및 특징

- 시스템이 제공하는 서비스와 외부 환경과의 관계를 표현하는 Diagram
- 시간과 순서 개념없이 정적인 관점에서 모델링
- 시스템과 사용자 간의 상호 교류에 바탕을 둠
- 사용자 요구사항 분석 및 애플리케이션 기능 설계에 사용됨

구성 요소

액터	시스템 외부에 존재하며 시스템과 교류하는 사람이나 시스템
유스 케이스	시스템이 제공하는 서비스 혹은 기능
관계	액터와 유스케이스 사이에 상호작용하는 관계

UML Diagram의 유형

Class Diagram

정의 및 특징

- 클래스와 클래스 간의 관계를 나타내며 UML 모델 가운데 가장 공통적으로 많이 쓰이는 Diagram
- 객체지향 SW시스템 분석·설계에 사용되는 핵심적인 모델

관계(Relation)

- 두 클래스 간의 관계

Aggregation	Composition
• 두 클래스 간에 전체-부분의 관계, 전체와 부분 간에 생명주기가 다름	• Aggregation과 유사하지만 생명주기가 동일함

UML Diagram의 유형

Sequence Diagram

정의 및 특징

- 외부의 특정한 처리 요청을 해결하기 위해 필요한 객체들과 그 객체들이 참여한 시간적, 순서적 처리 흐름을 표현하는 Diagram
- Class 내부의 메소드 호출과 비즈니스 프로세스 설계도 검증 가능

구성 요소

- Things

  액터(Actor) 객체(Object)

- Relationships

  메시지(Message)

- 기타

  Life Line Focus of Control

UML Diagram의 유형

Component Diagram

정의

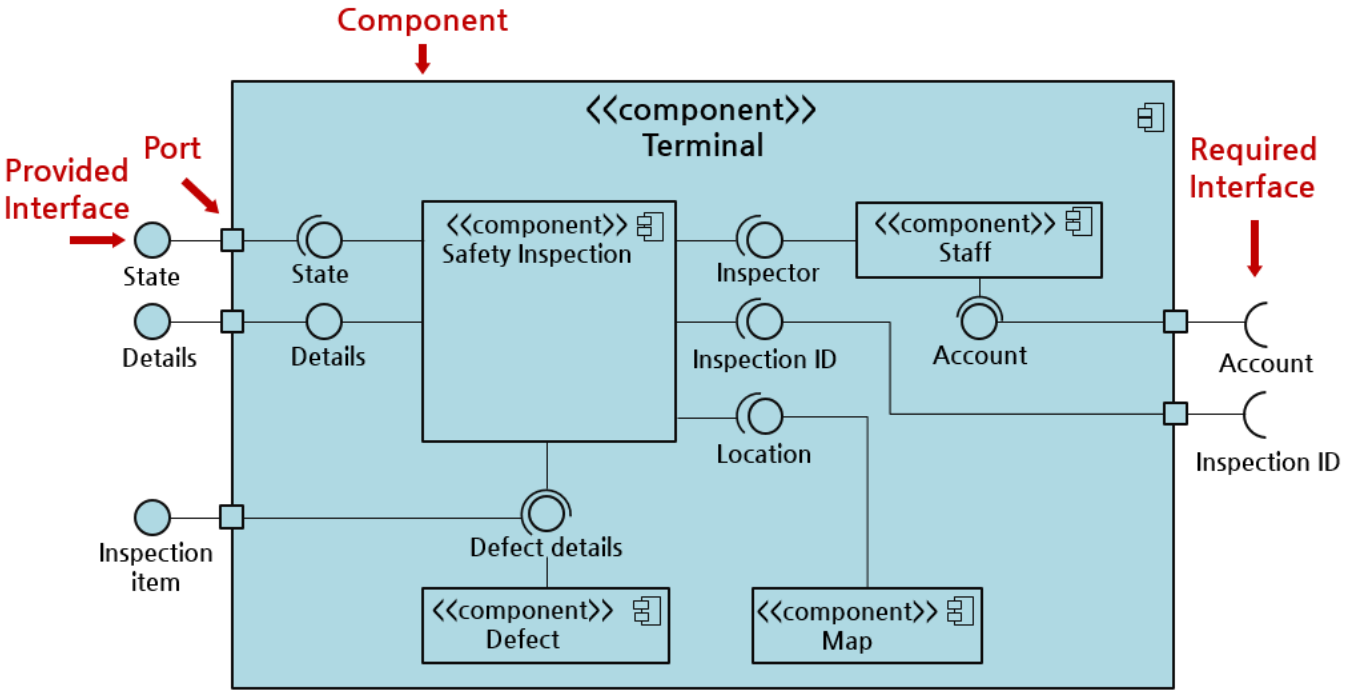
- 여러 개의 Class들을 묶고, 인터페이스를 통해 하나의 단위 업무처리를 할 수 있도록 만든 Diagram

구성 요소

Provided Interface	Required Interface
• 컴포넌트가 제공하는 인터페이스	• 컴포넌트가 기능 수행을 위해 외부에 필요로 하는 인터페이스

UML Diagram의 유형

Component Diagram



학습정리

1. 애플리케이션 설계 방법론의 개요

- 애플리케이션 설계 방법론은 개발에 필요한 여러 작업 단계의 수행방법(Method)과 기법(Technique) 및 도구(Tool)들을 체계적으로 정리한 규범

2. 애플리케이션 설계 방법론의 유형

- 애플리케이션 설계 방법론은 1960년대 구조적 방법론, 1990년대 정보공학 방법론, 2000년대 객체지향 방법론, 2005년 CBD, SOA, MDD로 발전해 옴
- 최근에는 MSA(Micro Service Architecture), DevOps, CI/CD 등 기술 발전과 함께 Agile/XP/Kanban방법론이 사용되고 있음

학습정리

3. UML의 개요

- UML : 객체 기술에 대한 표준화 기구(OMG)에서 인정한 객체지향 분석/설계 모델링 언어
- UML은 가시화 언어, 명세화 언어, 구축 언어, 문서화 언어임
- UML은 다이어그램(Diagram), 사물(Things), 관계(Relationship)로 구성되어 있음

4. UML Diagram의 유형

- Use-Case Diagram은 시스템이 제공하는 서비스와 외부 환경과의 관계를 표현하는 Diagram임
- Class Diagram은 클래스와 클래스 간의 관계를 나타내며 UML 모델 가운데 가장 공통적으로 많이 쓰이는 Diagram임
- Sequence Diagram은 외부의 특정한 처리 요청을 해결하기 위해 필요한 객체들과 그 객체들이 참여한 시간적, 순서적 처리 흐름을 표현함
- Component Diagram은 여러 개의 Class들을 묶고, 인터페이스를 통해 하나의 단위업무처리를 할 수 있도록 만든 Diagram임