<1과목>

2020-1회차

GoF (Gang of Four)

* 생성패턴 : 객체의 생성과 관련된 패턴

1. Builder : 작게 분리된 인스턴스를 건축하듯이 조합하여 객체를 생성
2. Prototype : 원본 객체를 복제하는 방법으로 객체를 생성
3. Abstract factory
4. Singleton

* 구조패턴 : 클래스나 객체들을 조합하여 더 큰 구조로 만들 수 있게 해주는 패턴

1. Bridge : 구현부에서 추상층을 분리하여, 서로가 독립적으로 확장할 수 있도록 구성
2. Deployment
3. adapter

* 행위패턴 : 클래스타 객체들이 서로 상호작용하는 방법이나 책임 분배 방법을 정의하는패턴

1. 유스케이스(UseCase) 다이어그램  
2. 시퀀스(Sequence) 다이어그램  
3. 커뮤니케이션(Communication) 다이어그램  
4. 상태(State) 다이어그램  
5. 활동(Activity) 다이어그램  
6. 상호작용 개요(Interaction Overview) 다이어그램  
7. 타이밍(Timing) 다이어그램

UML 모델

|  |
| --- |
| 기능적 모델은 사용자 측면에서 본 시스템 기능이며, UML에서는 Use case Diagram을 사용한다.  정적 모델은 객체, 속성, 연관관계, 오퍼레이션의 시스템의 구조를 나타내며, UML에서는 Class Diagram을 사용한다.  동적 모델은 시스템의 내부 동작을 말하며, UML에서는 Sequence Diagram, State Diagram, Activity Diagram을 사용한다. |

* 시간에 독립적이고 시스템의 개념과 어떻게 각각이 연결되어 있는지를 전달하는 방법
* UML의 구성요소로는 사물, 관계, 다이어그램 3가지로 이루어져있으며,  
  Things은 사물, Relationship은 관계, Diagram은 다이어그램입니다.
* 시퀀스 다이어그램 구성

-> 객체 사이에 오가는 메시지를 시간순으로 나타냄

1. 액터

2. 활성 객체(object)

3. 라이프라인(생명선)

4. 메세지

5. 제어 삼각형

* 구조 다이어그램

1. Class 클래스
2. Object 객체
3. Composite Structure 복합체 구조
4. Deployment 배치
5. Component 컴포넌트
6. Package 패키지 다이어그램

객체지향 기법

* 집단화 is part of(part가 들어가면 집단화 생각) : 클래스 간의 구조적인 집약 관계 "클래스 A는 클래스 B와 클래스 C로 구성된다"
* 일반화 is a :클래스들 간의 개념적인 포함 관계  "자식 클래스 A는 부모 클래스 B의 일종이다."
* 캡슐화 : 속성 (데이터)과 메소드(연산) 을 하나로 묶어서 객체로 구성된다.
* 추상화 : 공통 성질을 추출하여 수퍼클래스로 구성한다. 또한 객체 중심의 안정된 모델을 구축 가능 하며 현실 세계를 자연스럽게 표현한다. 장점으로 분석의 초점이 명확해진다.

S/W 설계시 구축된 플랫폼의 성능특성 분석

* 응답시간
* 가용성
* 사용률
* 정확성 \* 응가사정 으로 외우자 \*

자료사전 기호

* 정의 =
* 구성, 연결 +
* 반복 {}
* 주석 \*\*
* 선택 [|]
* 생략 ()

특성

직관성 : 누구나 쉽게 이용하고 쉽게 사용할 수 있어야 함  
유효성 : 정확하고 완벽하게 사용자의 목표가 달성될 수 있도록 제작  
학습성 : 초보와 숙련자 모두가 쉽게 배우고 사용할 수 있게 제작  
유연성 : 사용자의 인터랙션을 최대한 포용하고, 실수를 방지할 수 있도록 제작

XP 5원칙

* 용기
* 단순성
* 커뮤니케이션
* 피드백
* 존중

[Case가 가지는 주요 기능]  
<=>Case란 소프트웨어 개발의 자동화<=>

1.S/W의 생명주기 전(모든)단계의 연결  
2.모델들 사이의 모순검사  
3.오류검증  
4.자료흐름도 등 다이어그램 작성  
5.다양한 소프트웨어 개발 모형지원  
6.시스템 문서화 및 명세화를 위한 그래픽 지원

DBMS 고려사항

1.무결성(가용성)  
2.일관성(상호호환성)  
3.회복  
4.보안  
5.효율성(성능)  
6.데이터베이스 확장

객체지향 분석 방법론

Rumbaugh - 가장 일반적으로 사용되는 방법으로 분석 활동을 객체/동적/기능 모델로 나누어 수행하는 방법  
Booch - 미시적(Micro) 개발 프로세스와 거시적(Macro) 개발 프로세스를 모두 사용하는 분석방법  
Jacobson - Use Case를 강조하여 사용하는 분석방법  
Coad와 Yourdon - E-R다이어그램을 사용하여 개체의 활동들을 데이터 모델링하는데 초점을 둔 기법  
Wirfs-Brock - : 분석과 설계간의 구분이 없고 고객 명세서를 평가해서 설계 작업까지 연속적으로 수행하는 기법

DFD

시간 흐름을 명확하게 표현할 수 없다  
자료에 대한 흐름을 표현하며 구조적 분석기법에 이용된다.  
요소는 화살표, 원 , 직선으로 표시한다.

Data flow, data store, terminal, process

상위 설계 : 아키텍처 설계, 데이터 설계, 시스템 분할, 인터페이스 정의, 사용자 인터페이스 설계(UI 설계)  
  
하위 설계 : 모듈 설계, 인터페이스 작성

코드(Code)의 주요 기능  
- 식별 기능 : 데이터 간의 성격에 따라 구분  
- 분류 기능 : 특정 기준이나 동일한 유형에 해당하는 데이터를 그룹화  
- 배열 기능 : 의미를 부여하여 나열  
- 표준화 기능 : 다양한 정보를 컴퓨터에 의해 처리하기 위해서는 단일한 형태의 표준화된 표현이 필요함

객체지향 설계원칙  
  
1. 단일 책임 원칙(SRP, Single Responsibility Principle)  
객체는 단 하나의 책임만 가져야 한다.  
  
2. 개방-폐쇄의 원칙(OCP, Open Closed Principle)  
기존의 코드를 변경하지 않으면서 기능을 추가할 수 있도록 설계가 되어야 한다.  
  
3. 리스코프 치환 원칙(LSP, Liskov Substitution Principle)  
일반화 관계에 대한 이야기며, 자식 클래스는 최소한 자신의 부모 클래스에서 가능한 행위는 수행할 수 있어야 한다.  
  
4. 인터페이스 분리 원칙(ISP, Interface Segregation Principle)  
인터페이스를 클라이언트에 특화되도록 분리시키라는 설계 원칙이다.  
  
5. 의존 역전 원칙(DIP, Dependency Inversion Principle)  
의존 관계를 맺을 때 변화하기 쉬운 것 또는 자주 변화하는 것보다는 변화하기 어려운 것, 거의 변화가 없는 것에 의존하라는 것

요구사항 명세기법

정형 명세법  
- 수학적 기반/모델링 기반  
- Z, VDM, Petri-Net(모형기반)  
- CSP, CCS, LOTOS(대수적방법)  
- 시스템 요구특성이 정확하고 명세가 간결하다. 명세와 구현이 일치.  
- 그러나 이해도가 낮으면 이해관계자의 작성 부담 가중.  
  
비정형명세  
- 상태, 기능, 객체 중심 명세법  
- FSM(Finite state machine)  
- Decision Table, ER모델링  
- State chart(SADT)  
- UseCase : 사용자기반모델링  
- 명세 작성이 간편하고 의사전달 방법이 다양하다.  
- 불충분한 명세가능성, 모호성.

미들웨어

1. WAS : 애플리케이션 수행 미들웨어  
2. MOM : 메세지 지향 미들웨어  
3. RPC : 원격 프로시저 호출  
4. ORB : 네트워크 호출 미들웨어