14주차 1차시 객체지향 설계의 구체적 절차

【학습목표】

1. 분석론의 기초개념 및 원칙을 설명할 수 있다.
2. 객체지향 설계의 구체적 절차를 구분할 수 있다.

학습내용1 : 분석론 기초사항 고찰

1. 기초개념 및 원칙

1) 설계의 대상
- 객체지향 분석단계는 주로 시스템이 「무엇(what)」을 수행할 것인지 중점을 두었고, 객체지향설계단계는 주로 연산·기능이 「어떻게(how)」수행해야 되는지 정의하여, 객체를 구현하는데 필요한 내용을 구체적으로 결정함.

- 객체지향 설계단계에서 「객체에 대하여 결정해야 할 사항」은 다음과 같다.
    - 다른 객체와 메시지 교환에 필요한 사항을 결정하는 프로토콜 기술을 해야 함.
    - 연산에 필요한 객체의 내부활동을 정의하는 구현 기술(implementation description)을 해야 함.

- 특정업무에 관련된 「독립적인 객체들을 유기적으로 연결」 하기 위해서는 객체들을 동적으로 연결하는 유일한 수단인 「메시지의 구조를 정의하는 작업」 이 「프로토콜 기술(Protocol Description)」임. 이 프로토콜 기술과정에서 객체가 수행할 연산의 결정, 메시지 구조 결정, 인터페이스 등이 결정되어야 함.

- 객체지향 기법을 적용한 시스템은 독립적인 객체 사이에 정보의 교환은 메시지를 통해서만 가능하기 때문에 객체는 메시지를 정확히 교환 가능한 인터페이스(interface)를 가지고 있어야, 다른 객체의 연산처리 결과에 정확하게 접근할 수 있음.

- 특정한 객체가 메시지를 전송하여 원하는 객체의 서비스를 받기 위해서는 해당 객체를 호출하는 기능을 가지고 있어야 함.
- 이때 해당 메시지의 내용과 구조는 상대방의 그것과 일치해야 함.

- 구현기술(implementation description) 정의에는 메시지에서 지정한 연산의 수행에 요구되는 모든 내용을 결정하므로 속성을 나타내는 내부자료 구조의 결정, 연산절차와 알고리즘의 결정, 속성 및 연산의 상속구조 결정 등이 이루어져야 함.

- 객체를 역할이라는 측면에서 볼 때 메시지를 전송하는 객체는 수신 객체에서 처리한 정보를 사용하는 「소비자(consumer)」로 볼 수 있고, 메시지를 받은 수신객체는 송신객체에서 요구한 새로운 정보를 생성하여 제공하는 「생산자(producer)」로 볼 수 있음.
- 결국 객체는 그 역할에 따라서 「생산자」와 「소비자」 로 구분됨.

2. 객체지향 설계의 과정

1) 객체지향 설계의 과정
- 시종 객체를 중심으로 작업이 진행됨
- 그러므로 분석단계 의 기본적 사상이 설계·구현단계로 이어져 분석단계에서 누락 · 잘못된 내용을 수정·보완하는데 유리하다는 장점을 가지기도 함.
- 그러나 이 장점은 분석단계와 설계단계를 명확히 구분할 수 없다는 단점이기도 함.

2) 설계된 결과물에 대한 문서화는 누구나 용이하게 이해·
- 사용하기 위해서는 「표준화 된 표기법(notation)」으로 정리해야 함.

3) 복잡하고 거대한 시스템에 대하여 구체적이고 상세한 내용들을 하나의 독립된「뷰(view)」로 취급은 불가능함.
- 그래서 발생되는 행위에 대해서 이해하기 쉽고, 시스템의 행동방식과 구조를 판단하기 쉽다는 이유로「멀티 뷰(multi view)」를 사용함.

4) 이상과 같은 관점에서 시스템을 모델화 하는 과정에서 다음과 같은 다양한 다이어그램을 사용함.

① 논리적 모델(logical model) : 시스템의 논리적 뷰(view)를 통하여 「시스템의 구조를 나타내는 도표」를 표현하는 도구로서 클래스 다이어그램, 객체 다이어그램 등을 사용함.

② 물리적 모델(physical model) : 물리적 뷰(view)를 통하여 시스템의 실질적인 행동 양식을 나타내는 도표를 표현하는 도구로서 모듈 다이어그램, 프로세스 다이어그램 등을 사용함.

- 시스템이 실질적으로 작용하는 경우에 동적 행위를 포함한 시스템의 구조 및 행동 양식을 나타내려면 상태 다이어그램을 사용해도 됨.

3. 설계 결과물의 평가기준

1) 설계의 결과물에 대한 평가기준은 반드시 존재해야 함.
- 우수한 설계 결과물은 기본적으로 개발·유지보수가 용이해야 한다는 전제를 만족시켜야 함
- 그러므로 다음 조건을 만족시키는 특징을 가져야 함.
- 복잡·거대한 문제를 쉽게 해결하기 위하여 「분할을 지원하는 기능 (decomposability)」을 구비해야 함.
- 설계·구축된 모듈의 「재 사용성 정도(composability)」가 높아야 함.
- 다른 모듈·정보의 참조 없이 이해되게 「이해의 용이성(understandability)」이 높아야 함.
- 특정 모듈에 오류 발생 시에 「부작용 확산을 최대한 억제하는 기능(protection)」을 가져야 함.
- 모듈 내부 변경으로 인해서 「관련 모듈에 매핑(mapping) 되는 변경을 명확히 해 주는 기능(continuity)」을 가져야 함.

2) 기존의 모듈화 프로그래밍(modular programming) 기법에도 위에서 언급한 기능을 가지 고 있음.
* 그러므로 그들 기능 중에 객체지향 설계에 적용 가능한 내용을 요약하면 다음과 같음.
- 모듈은 사용언어를 이용하여 구문적 단위로 매핑 가능할 경우 「언어의 모듈러 단위(linguistic modular unit)」로 정의 할 수 있고, 이는 사용하려는 프로그래밍 언어는 정의된 모듈화를 직접지원 가능해야 한다는 것임.

* 예를 들어 서브루틴이 필요한 경우에는 일반적 언어를 이용하여 구문단위로 구현 가능한데 비하여 자료구조나 프로세스를 포함하여 하나의 단위인 패키지(package)로 정의된 경우에는 「객체지향 언어」가 사용 되어야 함.

- 인터페이스로 교환되는「정보의 량」을 「최소화(few interface)」시켜야 함
- 모듈 사이의「인터페이스 수(數)」를 「최소한(small interface)」으로 하여 「낮은 결합도(week coupling)」를 유지해야 함.
- 모든 정보는 객체의 인터페이스를 통하여 「명시적(explicit interface)」으로 수수(授受)되어야 함.
- 「공개정보(public information)」로 정의되지 않는 한 모든 모듈 정보는 「내부에 은폐(information hiding)」되어야 함.

학습내용2 : 객체지향 설계의 구체적 절차

1. 객체지향 설계의 구체적 절차

1) 객체지향 설계의 절차·기준은 사람에 따라서 다소의 차이를 보이지만 OMT 제안에 의거하여 다음과 같이 8 단계로 구분하여 고찰함.
- 시스템 분할
- 동시성 문제 해결
- 처리기 할당
- 자료관리법 확정
- 자원 선정
- 소프트웨어 제어방법 결정
- 경계 조건 결정
- 우선순위 결정

2. 시스템 분할

1) 시스템이 복잡·거대한 경우 몇 개의 서브시스템으로「분할(partition)」해야 함.
- 개발·유지보수 작업을 용이하게 하기 위해서임.
- 분할함으로써 다수의 설계자가 동시에 참여하여 개발기간의 단축을 가져옴.

2) 시스템에 사용되는 하드웨어 처리기의 대수가 제한되기 때문에 특정업무에 관련되는 모든 객체를 동시에 수행이 불가능한 경우가 허다함.

3) 그러나 시스템이 실제로 실행될 경우
- 동시에 수행하지 않으면 안될 객체들은 어떤 방법으로라도 「동시성(concurrent) 문제」를 해결해야 함.

4) 「상태 다이어그램」을 분석하면 「객체 사이의 동시성」이 존재하는지 여부를 파악 가능함.
- 왜냐하면 이 다이어그램은 각 객체에 대하여 「동적모델」을 나타내고 있기 때문임.

3. 동시성 문제 해결

1) 동시성을 고려할 필요가 있을 경우에는 다음과 같이 대처함.
- 동시수행이 요구되는 객체들을 별도의 하드웨어 처리기를 배정하여 처리함.
- 단일 하드웨어 시스템에서 동시성 문제는 소프트웨어적 기능인 타스킹(tasking)·인터럽트(interrupt)를 이용하여 해결함.
- 일괄처리 시에 발생하는 동시성의 문제는 멀티타스킹 기법으로 해결함.
- 비 상호작용 성 사건(event)을 동시에 받아들이는 객체들은 동시에 수행할 수 있는 대책을 수립해야 함.

4. 처리기 할당

1) 사용자의 요구를 충족시킬 수 있도록 처리할 정보의 량과 처리시간을 고려하여 다음 과 같은 사항을 생각 해야 함.

① 처리기의 처리능력과 필요한 처리기의 대수(臺數)
② 처리기에 특정한 성능이 요구될 경우 전용 하드웨어로 해결 가능성 여부 조사(불가능 시에 소프트웨어적인 대응방안 모색)
③ 원격지에 격리되어 처리하는 타스크는 별도 처리기에 배정한다는 원칙 준수 여부
④ 상호 작용성 서브시스템들의 동일 처리기에 배정 원칙의 준수 여부

5. 자료관리법 확정

1) 분석단계에서 작성된 자료흐름도(DFD)에 나타난 「자료저장소(data store)의 내용」을 파일(file), 데이터베이스(database) 중에 어느 방법으로 구축할 것인가를 결정하는 것을 의미함.

6. 자원 선정

1) 시스템 실행을 위해서 갖추어야 할 처리기, 워크스테이션, 하드디스크, 통신장비, 기타 장비 등의 선정과 관리운용에 대한 원칙을 수립해야 함

7. 소프트웨어 제어방법의 결정

1) 소프트웨어 제어방법은 객체 사이에서 발생하는 상호작용을 어떤 방식으로 소프트웨어가 구현하는가를 의미함.
- 이에는 다음과 같이 3 가지 방안이 있음.

2) 소프트웨어 제어방법의 결정
① 절차중심 시스템(procedure-driven system)
- 제어권을 프로그램이 가지면서 다른 프로그램을 호출하기 때문에 호출 시의 상태 도 프로그램 자신이 유지관리 하는 특징을 가짐.
- 전통적 프로그래밍 언어(C++, Smalltalk)로 구현 가능하다는 장점을 가짐.
- 구현 내용이 순차적으로 처리되므로 객체들의 동시성을 해결할 수 없다는 단점 있음.
- 피호출 프로그램에서 원하는 내용의 입력을 전달받을 경우에 자신이 관리하고 있는 호출 당시의 상태에서부터 처리를 계속해 나갈 수 있음.

② 사건 중심 시스템(event-driven system)
- 제어권을 OS나 프로그래밍 언어에서 제공하는 분배기(dispatcher)가 가지고 있으면서 발생하는 사건을 분석하여, 그 사건에 필요로 하는 프로그램을 호출하는 방식임. (X-Window 및 Sunview가 예임)
- 처리가 일시적으로 중단된 프로그램의 제어권은 일단 분배기로 돌려주기 때문에 프로그램의 상태는 항상 분배기에 의해서 유지관리 됨.
- 필요한 내용을 구현하기 어렵다는 단점을 가짐.
- 다른 방법에 비하여 일단 구현된 내용은 모듈화가 우수하여 좋은 시스템을 구축 할 수 있다는 장점을 가짐.

③ 동시적 시스템(concurrent system)
- UNIX처럼 다수의 타스크(task)를 조정할 수 있는 큐(queue)를 가진 OS에서 적용 되 는 방법임.
- 제어권은 독립된 객체가 가지고 있기 때문에 특정한 타스크가 실행을 중단하고 입력을 대기하고 있는 상태에서도 다른 타스크에서는 계속적인 실행이 가능하다는 특징을 가짐.
- 이 방법은 OS가 달라지면 호환성에 문제가 있음.

8. 경계조건의 설정

1) 시스템의 경우 사용자 오류, 프로그램 오류, 시스템 자원 제한 등의 비 정상적인 요인들로 발생하는 문제가 시스템의 다른 부분에 미치는 영향을 충분히 고려하여 설계해야 함.

2) 이 때 시스템의 초기화나 종료를 위하여 필요한 처리 내용을 명확히 정의해야 함.

9. 우선순위 결정

1) 시스템 개발과정에서 관련되는 여러 가지 요소 중에서 상호 배타적 혹은 트레이드 오프(trade off) 관계인 경우가 허다함.

2) 최적화 시스템을 구축하기 위해서는 이들 요인 사이에 우선순위(priority)를 결정하여 설계해야 함.

【학습정리】

1. 분석론 기초사항 고찰한다.

2. 객체지향 설계의 구체적 절차를 파악한다.

3. 소프트웨어 제어방법의 결정을 알아본다.