정보처리기사	정보처리기사 실기	교수 : 손승호
		과정 : 기본이론

제1장 요구사항 확인

제1절 현행 시스템 분석

- 개발하고자 하는 응용소프트웨어에 대한 이해를 높이기 위해, 현행 시스템의 적용현황을 파악함으로써 개발범위와 향후 개발될 시스템으로의 이행방향성을 분석할 수 있다.
- 개발하고자 하는 응용소프트웨어와 관련된 운영체제, 데이터베이스관리시스템, 미들웨어 등의 요구사항을 식별할 수 있다.
- 현행 시스템을 분석하여, 개발하고자 하는 응용소프트웨어가 이후 적용될 목표시스템을 명확하고 구체적으로 기술할 수 있다.

1. 현행 시스템 분석

(1) 현행시스템 파악의 개념

① 현행 시스템이 어떤 하위 시스템으로 구성되어 있는지, 제공하는 기능이 무엇인지, 다른 시스템들과 어떤 정보를 주고받는지를 파악하는 것이다. 또한 어떤 기술요소를 사용하고 있는지, 사용하고 있는 소프트웨어 및 하드웨어는 무엇인지, 네트워크는 어떻게 구성되어 있는지 등을 파악하는 활동이다.

② 현행 시스템 파악의 목적은 이를 통하여 향후 개발하고자 하는 시스템의 개발범위 및 이행방향성 설정에 도움을 주는 것이다.

(2) 현행 시스템 파악 절차

① 1단계

시스템의 구성, 기능, 인터페이스 현황을 파악하는 단계

㉠ 시스템의 구성 현황

- 현행 시스템 구성 현황은 조직의 주요 업무를 처리하는 기간 업무와 이를 지원하는 지원 업무로 구분하여 기술한 것이다.
- 각 업무에 속하는 단위 업무 정보시스템들의 명칭, 주요 기능들을 명시함으로써 조직 내 존재하는 모든 정보시스템의 현황을 파악하도록 한다.

㉡ 시스템의 기능 현황

- 기능 현황의 정의 : 단위 업무 시스템이 현재 제공하고 있는 기능을 기술한 것이다.
- 기능 현황 작성 시 고려 사항 : 단위 업무 시스템에서 제공하는 기능들을 주요 기능과 하부 기능으로 구분하여 계층형으로 표시한다.

[정보시스템 기능 구성도]

시스템명	기능 L1	기능 L2	기능 L3	비고
단위 업무 A 시스템	기능 1	하부기능 11	세부기능 111	
			세부기능 112	
		하부기능 12	세부기능 121	
			세부기능 122	
	기능 2	하부기능 21	세부기능 211	

㉢ 시스템 인터페이스 현황

- 인터페이스 현황의 정의 : 단위 업무 시스템이 다른 단위 업무 시스템과 주고받는 데이터의 종류, 데이터 형식, 프로토콜, 연계유형, 주기 등을 명시한 것이다.
- 인터페이스 현황 작성 시 고려 사항 : 중요한 고려 사항으로는 어떤 형식(format)으로 데이터를 주고받는지(XML, 고정 포맷, 가변 포맷 등), 어떤 통신규약(TCP/IP, X.25 등)을 사용하고 있고, 연계유형(EAI, FEP등)은 무엇인지 등이 있다.

[시스템 인터페이스 현황]

송신시스템	수신시스템	연동데이터	연동형식	통신규약	연계유형	주기
A 시스템	대외기관1	입고정보	XML	X.25	FEP	수시
A 시스템	대외기관2	출고정보	XML	TCP/IP	EAI	일

② 2단계

아키텍처 및 소프트웨어 구성 현황을 파악하는 단계

③ 3단계

현행 시스템의 하드웨어 및 네트워크 구성 현황을 파악하는 단계

2. 소프트웨어 수명 주기

(1) 소프트웨어의 생명주기 개요

① 소프트웨어 제작 공정 과정이다.
② 시스템 개발주기(System Development Life Cycle : SDLC)라 부른다.
③ 개발 단계에서 점차 변화해 가면서 나오는 소프트웨어 형상(Configuration)을 가시화한다.
④ 소프트웨어가 개발되기 위해 정의되고 사용이 완전히 끝나 폐기될 때까지의 전 과정이다.

(2) 폭포수형 모형(선형순차모형, 전형적인 생명주기 모형 : Boehm, 1979)

1) 폭포수형 모형 개요

① 소프트웨어의 개발 시 프로세스에 체계적인 원리를 도입할 수 있는 첫 방법론이다.
② 적용사례가 많고 널리 사용된 방법이다.
③ 단계별 산출물이 명확하다.
④ 각 단계의 결과가 확인된 후에 다음 단계로 진행하는 단계적, 순차적, 체계적인 접근 방식이다.
⑤ 기존 시스템 보완에 좋다.
⑥ 응용 분야가 단순하거나 내용을 잘 알고 있는 경우 적용한다.
⑦ 비전문가가 사용할 시스템을 개발하는 데 적합하다.

2) 폭포수형 모형 단계

① 계획 단계
- 문제를 파악하고 시스템의 특성을 파악하여 비용과 기간을 예측한다.
- 개발의 타당성을 분석하고 전체 시스템이 갖추어야 할 기본기능과 성능 요건을 파악한다.

② 요구 분석 단계
- 사용자 요구를 정확히 분석, 이해하는 과정으로 구현될 시스템의 기능이나 목표, 제약사항 등 정확히 파악한다.
- 소프트웨어의 기능, 성능, 신뢰도 등 목표 시스템의 품질을 파악하는 것 이다.
- 개발자(분석가)와 사용자간의 의사소통이 중요하며, 명확한 기능 정의를 해야 한다.

③ 설계 단계
- 요구사항을 하드웨어 또는 소프트웨어 시스템으로 분배하는 과정이다.
- 모든 시스템의 구조를 결정하게 되는데, 소프트웨어 설계는 프로그램의 데이터 구조, 소프트웨어 구조, 인터페이스 표현, 알고리즘의 세부 사항들에 초점을 맞춰 진행한다.
- 한 개 이상의 실행 가능한 프로그램으로 변환할 수 있는 형태로 소프트웨어의 기능을 표현한 것이다.

④ 구현 단계
- 설계의 각 부분을 실제로 프로그래밍 언어를 이용하여 코드화하는 단계이다.
- 각 모듈(module) 단위로 코딩을 한다.

⑤ 시험 단계
- 각 프로그램 단위의 내부적으로 이상 여부 및 입력에 따라 요구되는 결과로 작동하는지의 여부를 확인한다.

⑥ 운용(operation) 및 유지보수(maintenance) 단계
- 사용자에게 전달되어 실제로 사용되며, 전달 이후에 발생하는 변경이 있다면 변경 요구를 수용하고 계속적인 유지를 해주어야 한다.

(3) 프로토타이핑 모형(Prototyping Model)

① 폭포수 모형에서의 요구사항 파악의 어려움을 해결하기 위해 실제 개발될 소프트웨어의 일부분을 직접 개발하여 사용자의 요구 사항을 미리 정확하게 파악하기 위한 모형이다.

② 순서 : 요구사항분석 → 신속한 설계 → 프로토타입 작성 → 사용자 평가 → 프로토 타입의 정제 (세련화) → 공학적 제품화

③ 장점

- 사전에 사용자의 요구사항의 신속하고 정확한 파악
- 시스템 개발 초기에 사용자가 개발에 참여함으로써 오류를 조기에 발견할 수 있다.

(4) 나선형 모형(spiral model)

① 폭포수 모델과 프로토타이핑 모델의 장점을 수용하고, 새로운 요소인 위험 분석을 추가한 진화적 개발 모델이다.

② 프로젝트 수행 시 발생하는 위험을 관리하고 최소화하려는 것을 목적으로 한다.

③ 계획수립, 위험분석, 개발, 사용자 평가의 과정을 반복적으로 수행한다.

(5) 애자일(Agile)

1) 애자일의 정의

① 애자일 소프트웨어 개발(Agile software development) 혹은 애자일 개발 프로세스는 소프트웨어 엔지니어링에 대한 개념적인 얼개로, 프로젝트의 생명주기동안 반복적인 개발을 촉진한다.

② 애자일 개발 프로세스란 어느 특정 개발 방법론을 가리키는 말은 아니고 "애자일개발을 가능하게 해 주는 다양한 방법론 전체를 일컫는 말이다.

③ eBusiness 시장 및 SW개발환경 등 주위변화를 수용하고 이에 능동적으로 대응하는 여러 방법론의 통칭한다.

2) 애자일의 특성

① Predictive라기보다 Adaptive(가변적 요구사항에 대응)

② 프로세스중심이 아닌 사람 중심(책임감이 있는 개발자와 전향적인 고객)

③ 전반적인 문서화보다는 제대로 작동하는 소프트웨어

④ 계약 협상보다는 고객 협력

⑤ 계획을 따르기보다는 변화에 응대함

⑥ 모든 경우에 적용되는 것이 아니고 중소형, 아키텍처 설계, 프로토타이핑에 적합하다.

3) 애자일의 종류

① 익스트림 프로그래밍(Extreme Programing, XP)

- 애자일 개발 프로세스의 대표자로 애자일 개발 프로세스의 보급에 큰 역할을 하였다.
- 이 방법은 고객과 함께 2주 정도의 반복개발을 하고, 테스트와 우선 개발을 특징으로 하는 명시적인 기술과 방법을 가지고 있다.

② 스크럼

- 30일마다 동작 가능한 제품을 제공하는 스플린트를 중심으로 하고 있다.
- 매일 정해진 시간에 정해진 장소에서 짧은시간의 개발을 하는 팀을 위한, 프로젝트 관리 중심의 방법론이다.

③ 크리스털 패밀리
- 이 방식은 프로젝트의 규모와 영향의 크기에 따라서 여러종류의 방법론을 제공한다.
- 그중에서 가장 소규모 팀에 적용하는 크리스털 클리어는 익스트림 프로그래밍만큼 엄격하지도 않고 효율도 높지 않지만, 프로젝트에 적용하기 쉬운 방법론이다.

④ Feature-Driven Development
- feature마다 2주정도의 반복 개발을 실시한다. Peter Coad가 제창하는 방법론으로써, UML을 이용한 설계 기법과도 밀접한 관련을 가진다.

⑤ Adaptive Software Development, ASD
- 소프트웨어 개발을 혼란 자체로 규정하고, 혼란을 대전제로 그에 적응할 수 있는 소프트웨어 방법을 제시하기 위해 만들어진 방법론이다.
- 내용적으로는 다른 방법론들과 유사하지만, 합동 어플리케이션 개발(Joint Application Development, 사용자나 고객이 설계에 참가하는 개발 방법론)을 사용하고 있는 것이 조금 다르다.

4) 익스트림 프로그래밍(eXtreme Programming, XP)
① 익스트림 프로그래밍(eXtreme Programming, XP)는 켄트 백 등이 제안한 소프트웨어 개발 방법이다.
② Agile Process의 대표적 개발기법이며, 비즈니스 상의 요구가 시시각각 변동이 심한 경우에 적합한 개발 방법이다.
③ 개발자, 관리자, 고객이 조화를 극대화하여 개발생산성을 높이고자 하는 접근법이다.

※ XP(eXtreme Programming)의 5가지 핵심 가치
① 존중(Respect) : 팀 기반의 활동 중 팀원 간의 상호 존중을 강조
② 단순성(Simplicity) : 사용되지 않는 구조와 알고리즘 배제
③ 의사소통(Communication) : 개발자, 관리자, 고객간의 원활한 의사소통
④ 피드백(Feedback) : 지속적인 테스트와 통합, 반복적 결함 수정, 빠른 피드백
⑤ 용기(Courage) : 고객의 요구사항 변화에 능동적인 대처

※ XP(eXtreme Programming)의 실천사항
① 점증적인 계획 수립 : 계획세우기 . 우선순위와 기술사항 고려 범위 결정
② 소규모 시스템 릴리스 : 짧은 사이클로 버전 발표
③ 시험우선개발 : 테스팅 . 단위 테스트를 계속 작성
④ 리팩토링
⑤ 페어pair 프로그래밍 : 가장 좋은 구현 방법 고민, 전략적인 방법 고민
⑥ 공동 소유권 : 개발자들 누구나 코드 수정
⑦ 지속적 통합
⑧ 유지할 수 있는 속도 : 1주에 40시간 작업
⑨ 현장의 고객 : 고객도 한 자리에
⑩ 표준에 맞춘 코딩

제2절 요구사항 확인

- 소프트웨어 공학기술의 요구사항 분석 기법을 활용하여 업무 분석가가 정의한 응용소프트웨어의 요구사항을 확인할 수 있다.
- 업무 분석가가 분석한 요구사항에 대해 정의된 검증기준과 절차에 따라서 요구사항을 확인할 수 있다.
- 업무 분석가가 수집하고 분석한 요구사항이 개발하고자 하는 응용소프트웨어에 미칠 영향에 대해서 검토하고 확인할 수 있다.

1. 요구분석기법

(1) 요구분석기법의 정의

① 요구사항 분석은 사용자의 요구사항을 명확히 규정하고, 시스템의 특성을 반영하는 과정이며, 이 단계에서 사용자의 뜻을 이해하고 업무를 분석한다.

② 사용자의 막연한 문제의식이나 요구로부터 시스템이나 소프트웨어의 목적, 수행할 작업 등을 요구조건으로 명세화한다.

③ 시스템의 목표를 확립하는 과정이며, 시스템이 만족시켜야 할 기능, 성능, 그리고 다른 시스템과의 인터페이스 등을 규명한다.

(2) 요구분석기법의 특성

① 기능 요구

- 사용자가 필요로 하는 정보처리 능력에 대한 것으로 절차나 입·출력에 대한 요구이다.
- 요구 기능이란 시스템SW가 반드시 수행해야 하거나 시스템SW를 이용하여 사용자가 반드시 수행할 수 있어야 하는 기능이다.

② 비기능 요구

- 비기능 요구사항이란 시스템SW의 동작에 필요한 특정 요구기능 외에 전체 시스템의 동작을 평가하는 척도를 정의하며, 안정성, 확장성, 보안성, 성능 등이 포함된다.

㉠ 성능(Performance)

- 명령에 대한 응답시간(Response Time)이나 데이터 처리량(Throughput) 등
- 주어진 하드웨어에서 나타나는 소프트웨어의 시간척도

㉡ 신뢰도(Reliability)

- 주어진 환경과 데이터와 명령에 믿을 수 있게 대응하는 능력 (정확성(Accuracy), 완벽성(Completeness), 견고성(Robustness))

㉢ 기밀 보안성(Security)

- 시스템으로의 불법적인 접근을 막고 기밀자료나 보안을 유지하기 위해 사용을 불허하는 소프트웨어 능력

㉣ 개발계획(Development plan)

- 개발 기간, 조직, 개발자, 개발 방법론

㉤ 개발비용(Cost)

- 사용자측에서 투자할 수 있는 한계

㉥ 환경(Environment)

- 개발·운용·유지보수 환경에 관한 요구

(3) 요구분석기법의 종류

① 회의

- 프로젝트와 관련된 여러 사람들이 한 장소에 모여서 의견을 교환하고 자료를 수집하는 방법이다.
- 주로 문제 정의, 해결 전략 수립, 사용자(부서)간 분쟁 조정 등을 목적으로 한다.
- 회의 주최자는 회의 목적을 분명히하고 시간내에 원하는 결과를 얻을 수 있도록 참여자들이 회의에 집중할 수 있도록 유도할 수 있어야 한다.
- 회의 주최자는 회의 전에 회의 목적, 참석자, 주요 안건, 시간 및 장소 등을 통보하고, 회의 후에는 회의록을 작성하여 회의 내용에 대해 참석자들의 확인을 받아 향후 발생할 수 있는 오해나 갈등을 미연에 방지한다.

② 인터뷰

- 회의가 다 대 다 소통 방법이라면 인터뷰는 주로 일 대 일 또는 일 대 다의 요구사항 분석 방법으로 상세 요구사항을 도출할 때 주로 사용된다.
- 효율적인 인터뷰를 위해서 사전에 인터뷰 가이드를 작성하고 인터뷰 후에는 인터뷰 노트를 작성하여 참여자의 확인을 받도록 한다.

③ 설문 조사

- 설문 조사는 불특정 다수에게 기본적인 요구 사항이나 특정 항목에 대한 소극적인 의견을 파악하기 위해 활용하는방법이다.
- 전체 구성원의 전반적인 성향을 확인하고 방향을 결정하기 위한 방법으로 설계에 직접 반영되거나 시스템 사양을 결정하는 직접 요인이 되지는 않는다.

④ 프로토타이핑

- 프로토타이핑이란 시스템의 전체 또는 일부에 대하여 실제로 운영하게 될 모델을 구성하는 방법으로 최종 시스템의 중요한 기능이나 특징을 사전에 테스트하여 시스템 설계, 개발 과정에서의 위험 요소를 최소화하고 사용자의 불명확한 요구 사항을 효과적으로 도출하기 위해 활용될 수 있다.

2. 요구공학

(1) 요구공학 개요

1) 정의

- 요구사항을 정의하고 문서화하는데 필요한 요구사항의 추출, 분석, 명세, 검증, 유지보수 및 관리의 제반공정에 대한 체계적 접근방법이다. (IEEE standard)

2) 요구공학의 특징

- 개발범위, 각종테스트 기준(단위, 통합, 인수), 감리, 검수 등 프로젝트 수행의 중요한 기준으로 활용된다.
- 사용자의 요구사항은 추상적이고 불분명하므로 분석이 필요하며, 지속적으로 변화하는 특성 가진다.

[요구사항 문제점 및 해결방안]

문제점	해결방안
이해부족	경험있는 인력 투입, 유즈케이스 모델링
의사소통부족	Workthrough, inspection, 워크샵, 의사소통 채널 단일화
표현의 어려움	모델링 기법(구조적 분석기법, 객체지향 분석기법)으로 가시화
요구사항 변경	변경 관리 계획, 유형별 분리

(2) 요구공학 프로세스

[요구공학 프로세스]

절차	내용	방법
요구사항 추출 (Elicitation)	기능적/비기능적 요구수집 과정	인터뷰, 워크샵(JRP, JAD), 설문조사, 브레인스토밍
요구사항 분석 (Analysis)	분석기법을 이용한 가능한 문제도출 및 요구사항 이해/정제하는 과정	객체지향분석(UML,모델링) 구조적분석(DFD, Data Dictionary)
요구사항 명세 (Specification)	분석된 요구사항의 문서화 과정	ER 모델링, FSM, 구조적 분석과 설계 기술(SADT)
요구사항 검증 (Validation)	명세화된 요구사항 검증과정	Review, Inspection, Walk-through
요구사항 유지보수 (Maintenance)	요구사항 신규발생, 변경의 체계적 관리 활동	Baseline 관리로 가시성, 추적성의 형상관리

(3) 요구사항 명세 기준

[요구사항 명세 속성]

명세속성	설명
정확성	요구사항은 정확해야 한다.
명확성	단 한가지로 해석되어야 한다.
완전성	모든 것(기능, 비기능)이 표현되어야 한다.
일관성	요구사항 간 충돌이 없어야 한다.
수정용이성	요구사항의 변경이 가능해야 한다.
추적성	제안서 등을 통해 추적이 가능해야 한다.

[요구사항 명세기법]

구분	정형 명세	비정형 명세
기법	수학적 기반/모델링 기반	상태/기능/객체 중심 명세 기법
종류	Z, VDM, Petri-Net, CSP, LOTOS	FSM, Decision Table, ER 모델링, SADT, UseCase
장점	시스템 요구특성의 정확, 명세 간결	명세작성 이해 용이 의사전달 방법 다양성
단점	낮은 이해도, 이해관계자의 부담 가중	불충분한 명세 기능, 모호성

※ 요구 분석 명세서

1. 시스템의 개요 및 요약
   ① 기능 요약 ② 사용 목적
2. 개발 운용 및 유지 보수 환경
   ① 개발여건과 장비 ② 운용 절차 및 제약 조건 ③ 유지 보수 방법과 환경
3 자료 흐름도 입출력 양식, 명령어 요약
   ① 사용자 및 기타 시스템 시각에서의 시스템 특성
   ② 모니터 화면
4. 기능 요구사항
   ① 필요기능 기술과 미니 명세 ② 입력, 기능, 출력과의 관계
5. 성능 요구 사항
   ① 시간적 요구
   ② 효율성 요구
   ③ 기 타
6. 예외 조건 및 이의 처리
   ① 입력 자료 내부 값, 매체 등 ② 용량의 제한
   ③ 시스템 고장 ④ 운용자의 한계
7. 초기 제공 기능 및 우선순위
   ① 필요 기능과 바람직한 기능 ② 단계적 요구
8. 변경 및 개선 예정 사항
   ① 사전 고려 사항 ② 신 기종 구입 및 예산 초과 관련
9 인수 기준 및 문서화 표준
   ① 기능 및 성능 시험 ② 코딩 표준 제도 및 문서화 표준화
10. 설계 지침
   ① 분석시 파악한 내용으로서 설계에 도움을 줄 수 있는 내용
11. 참고 자료
   ① 사람과 조직 ② 각종 서류 및 참고 서적
12. 용어 해설
   ① 사용자와 개발자 양측의 원만한 이해 소통을 위하여

제3절 분석모델 확인

- 소프트웨어 공학기술의 요구사항 도출 기법을 활용하여 업무 분석가가 제시한 분석모델에 대해서 확인할 수 있다.
- 업무 분석가가 제시한 분석모델이 개발할 응용소프트웨어에 미칠 영향을 검토하여 기술적인 타당성 조사를 할 수 있다.
- 업무 분석가가 제시한 분석모델에 대해서 응용소프트웨어를 개발하기 위해 필요한 추가적인 의견을 제시할 수 있다.

1. 모델링 기법

(1) 개념 모델의 역할

① 실세계의 복잡한 문제에 대한 모델링이 소프트웨어 요구사항 분석의 핵심이며, 모델은 문제가 발생하는 상황에 대한 이해를 증진시키고 해결책을 설명한다.

② 개념 모델은 문제 도메인의 엔티티(entity)들과 그들의 관계 및 종속성을 반영한다.

(2) 개념 모델의 종류와 표기법

① 다양한 모델을 작성할 수 있으며, 대부분의 모델링 표기법은 UML을 사용한다.

② 개념 모델의 종류로는 Use Case(사용 사례) Diagram, 상태 모델(State Model), 데이터 흐름 모델(Data Flow Model), 객체 모델(Object Model), 목표기반 모델(Goal-Based Model), 데이터 모델(Data Model) 등이 있다.

2. UML(Unified Modeling Language)

(1) UML의 정의

① 시스템의 여러 다양한 특성을 표현할 수 있는 방법이 있으며, 객체지향 분석/설계 표현 방법에 대한 표준으로 받아들여지고 있다.

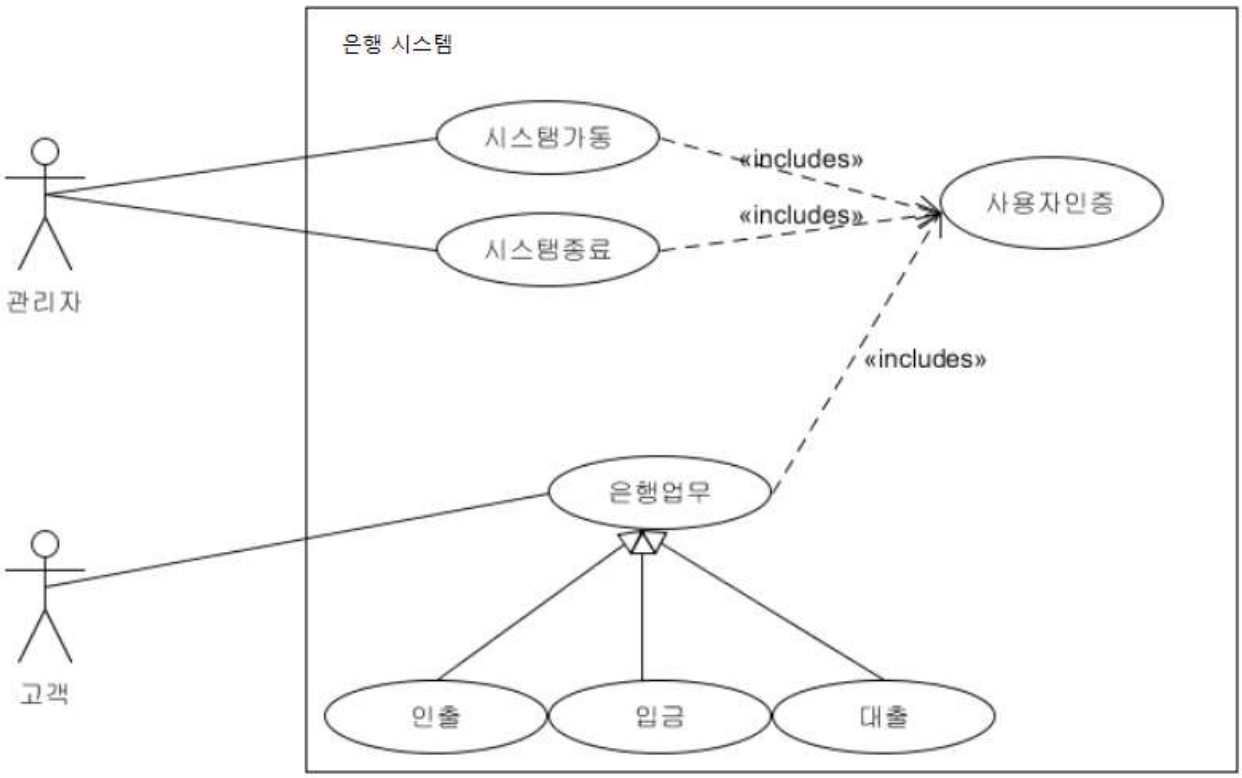
② 객체 지향 분석/설계용의 모델링 언어이며, 종래의 객체 지향 방법론과 함께 제안되어 모델링 언어 표기법의 표준화를 목적으로 한 것이다.

(2) UML의 종류

① Use Case(사용 사례) Diagram

- 시스템이 어떤 기능을 수행하고, 주위에 어떤 것이 관련되어 있는지를 나타낸 모형
- 외부에서 보는 시스템의 동작으로, 외부 객체들이 어떻게 시스템과 상호작용하는지 모델링한 것이다. (시스템이 외부 자극에 어떻게 반응하는가)
- 외부에서 보는 시스템의 동작에 초점을 두며 사용 사례(use case)와 액터(actor)로 구성된다.
- 액터는 시스템 범위 바깥쪽에 있고 사용 사례는 액터에게 보이는 시스템의 기능이다.

[Use Case 다이어그램]

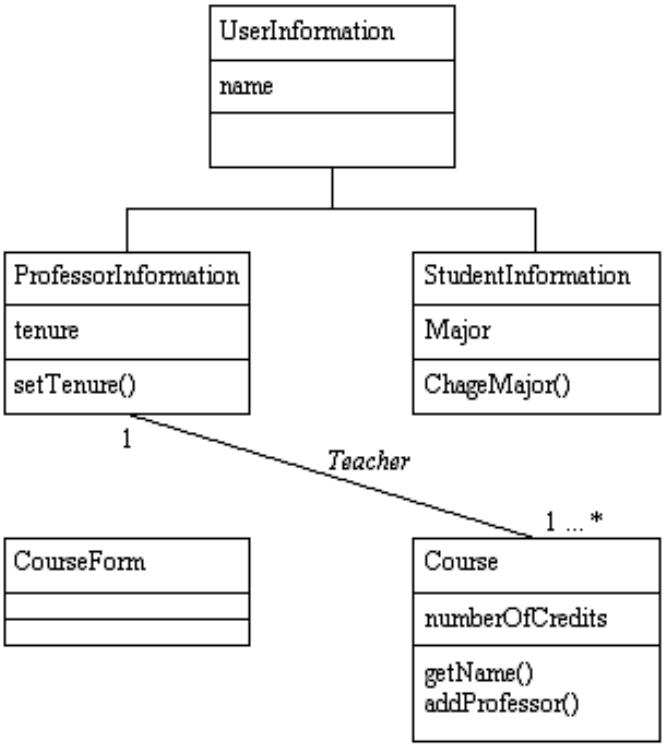


- 통신(communication) 관계 : 액터와 사용사례 사이의 관계를 선으로 표시하며 시스템의 기능에 접근하여 사용할 수 있음을 의미한다.
- 포함(inclusion) 관계 : 복잡한 시스템에서 중복된 것을 줄이기 위한 방법으로 함수의 호출처럼 포함된 사용사례를 호출하는 의미를 갖는다.
- 확장(extention) 관계 : 예외 사항을 나타내는 관계로 이벤트를 추가하여 다른 사례로 확장한다.
- 일반화(generalization) : 사용사례의 상속을 의미하며 유사한 사용사례를 모아 일반적인 사용사례를 정의한다.

② Class Diagram

- Class 다이어그램은 객체, 클래스, 속성, 오퍼레이션 및 연관관계를 이용하여 시스템을 나타낸다.
- Class 다이어그램을 통하여 사용자는 보다 쉽게 원하는 시스템의 구조를 정의할 수 있다. 또한 입출력 화면도 하나의 객체로 나타나기 때문에 시스템의 구조화가 용이하고 분석단계에서 사용자 인터페이스 프로토타이핑 작성이 쉬워진다.
- Class 다이어그램에서 클래스는 사각형으로 나타난다. 사각형은 다시 세 부분으로 나뉘는데 제일 위쪽은 객체명 중간은 객체의 속성, 아래쪽은 연산을 나타내게 된다.

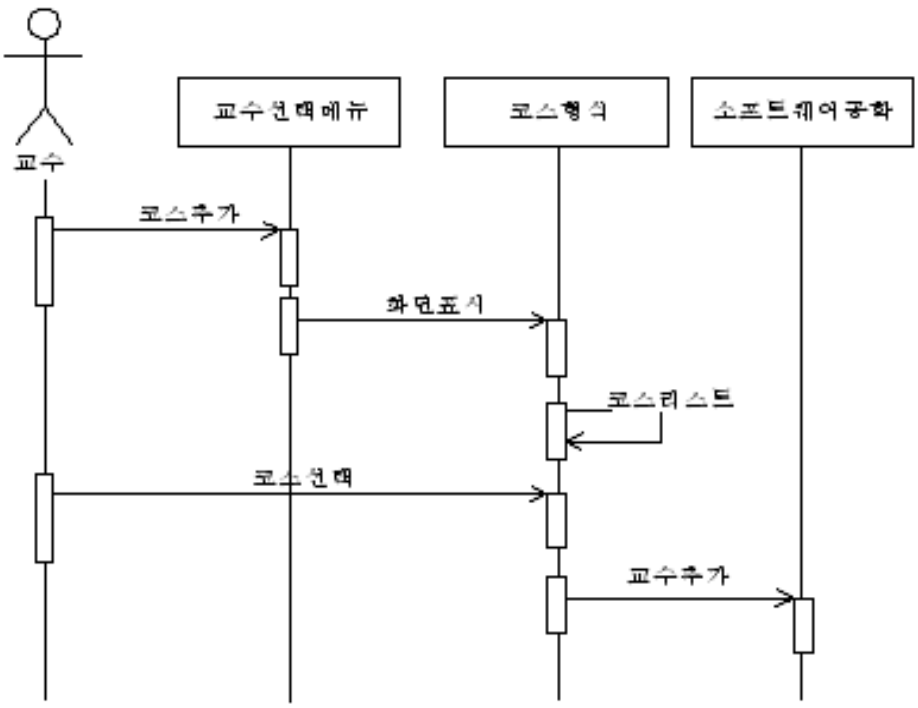
[학사에 관한 클래스 다이어그램]



③ Sequence Diagram

- 순서 다이어그램은 객체간의 메시지 통신을 분석하기 위한 것이다. 이는 시스템의 동적인 모델을 아주 보기 쉽게 표현하고 있기 때문에 의사소통에 매우 유용하다.
- 시스템의 동작을 정형화하고 객체들의 메시지 교환을 시각화하여 나타낸다.

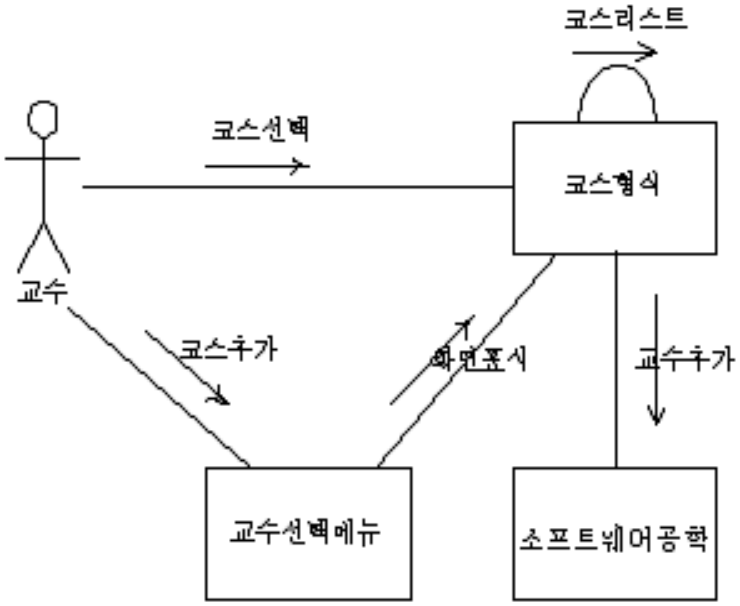
[강의등록의 Sequence 다이어그램]



④ Collaboration Diagram

- Sequence 다이어그램이 객체간의 메시지 처리에 대한 순서에 중점을 둔 반면, Collaboration 다이어그램은 관련 객체와의 연관성 분석에 중점을 두고 있다.
- UML의 다이어그램들 중에서 상호작용을 하는 객체들 사이의 조직을 강조하고, 시스템의 행동 관점으로 행동의 실제와 구현을 만들어내며, 클래스간의 상호작용 관계들을 통해서 메시지를 교환하는 작용을 나타내는 다이어그램이다.

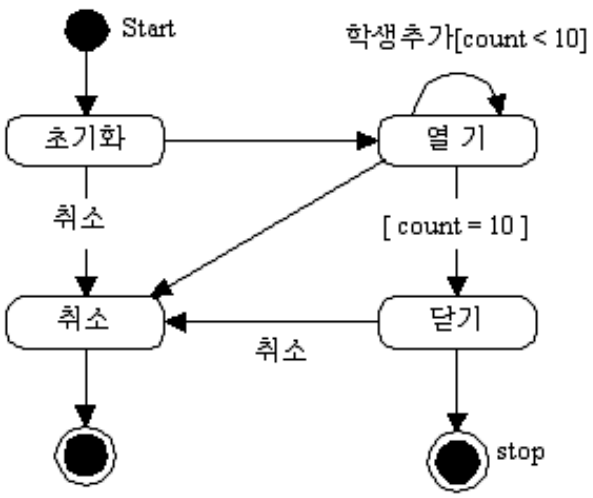
[강의등록의 Collaboration 다이어그램]



⑤ State Diagram

- State 다이어그램은 객체 내의 동적 행위를 모형화하기 위한 것으로, 복잡한 객체 혹은 객체 내부의 프로세스를 표현하고자 할 때 사용된다.
- State 다이어그램에서 상태는 둥근 사각형으로 상태의 흐름은 화살표로 표시된다.

[학생등록의 State 다이어그램]



⑥ Activity Diagram
- Activity 다이어그램은 State 다이어그램과는 달리 시스템의 흐름 전체를 파악하기 용이하도록 행위를 중심으로 흐름을 표현한 것이다.
- Activity 다이어그램은 현재 업무의 흐름 파악이 용이하다. 따라서 업무 흐름 지향적인 문제 영역에서 작성하는 것이 좋다.
- 시스템을 액티비티로 표현한 것으로 오퍼레이션의 집합이 수행됨을 나타내는 상태이다.

⑦ Component Diagram
- 시스템을 구성하는 실제 소프트웨어 컴포넌트 간의 구성체계를 기술하므로 아키텍처 표현에 우수하다.
- 컴포넌트 다이어그램은 각 컴포넌트를 그리고 컴포넌트 간의 의존성 관계를 화살표로 나타낸다.

제2장 데이터 입출력 구현

1. 데이터저장소

(1) 논리 데이터 모델링
- 사용자들의 요구 사항을 분석하여 DB에 저장될 정보를 파악하고, 필요한 정보들 간의 연관 관계를 모형화하는 과정이다.

① 데이터 모델링 절차

㉠ 요구 조건 분석
- 사용자가 원하는 데이터베이스의 용도를 파악하는 것.
- 사용자의 요구 조건을 수집하여 분석하고 정형적인 요구 조건 명세를 만든다.

㉡ 개념적 모델링
- 사용자들의 요구사항을 이해하기 쉬운 형식으로 간단히 기술하는 단계.
- 개념 스키마 모델링과 트랜잭션 모델링을 병행적으로 수행
- 개념 스키마 모델링 : 데이터의 조직과 표현을 중심으로 한 데이터 중심 설계
- 트랜잭션 모델링(처리 · 흐름 중심 설계) : 응용을 위한 데이터 처리에 주안점을 둔 처리 중심 설계

㉢ 논리적 모델링
- 개념적 설계에서 만들어진 구조를 구현 가능한 data 모델로 변환하는 단계.
- 논리적 데이터 모델링은 목표하는 DBMS가 구현되어 있는 환경과 특성까지는 고려하지 않고 해당 DBMS가 지원하는 데이터 모델에 적합하게 변환한다. 즉, DBMS에 종속적이라 할 수 있다.
- 논리적 설계 단계는 앞 단계의 개념적 설계 단계에서 만들어진 정보 구조로부터 목표 DBMS가 처리할 수 있는 스키마를 생성한다. 이 스키마는 요구 조건 명세를 만족해야되고, 무결성과 일관성 제약 조건도 만족하여야 한다.

㉣ 물리적 모델링
- 논리적 데이터베이스 구조를 내부 저장 장치 구조와 접근 경로 등을 설계.
- 물리적 데이터베이스의 기본적인 데이터 단위는 저장 레코드이다.
- 파일이 동일한 타입의 저장 레코드 집합이라면, 물리적 데이터베이스는 여러 가지 타입의 저장 레코드 집합이라는 면에서 단순한 파일과 다르다.
- 물리적 데이터베이스 구조는 데이터베이스에 포함될 여러 파일 타입에 대한 저장 레코드의 양식, 순서, 접근 경로, 저장 공간의 할당 등을 기술한다.

㉤ 데이터베이스 구현

※ 논리적 데이터 독립성 : DB의 논리적 구조의 변화에 대해 응용프로그램들이 영향을 받지 않는 능력을 말한다.

※ 3단계 스키마
- 스키마(schema)란 데이터베이스의 구조(개체, 속성, 관계)에 대한 정의와 이에 대한 제약 조건 등을 기술한 것으로 컴파일 되어 데이터 사전에 저장한다.
- 어떤 입장에서 데이터베이스를 보느냐에 따라 스키마는 다르게 될 수밖에 없다.
  (ANSI/SPARC 3 Level Architecture - 외부스키마, 개념스키마, 내부스키마)

① 외부 스키마
- 가장 바깥쪽 스키마로, 전체 데이터 중 사용자가 사용하는 한 부분에서 본 구조
  (사용자가 무엇을 사용하느냐에 따라 다름) - 서브스키마, 뷰 라고도 함
- 사용자 개개인이 보는 자료에 대한 관점과 관련
- 사용자 논리 단계(user logical level)

② 개념 스키마
- 논리적 관점에서 본 구조로 전체적인 데이터 구조(일반적으로 스키마라 불림)
- 범 기관적 입장에서 데이터베이스를 정의 (기관 전체의 견해)
- 조직 논리 단계(community logical level)
- 모든 데이터 개체, 관계, 제약조건, 접근권한, 무결성 규칙, 보안정책 등을 명세

③ 내부 스키마
- 물리적 저장장치 관점에서 전체 데이터베이스가 저장되는 방법 명세
- 실제로 저장되는 내부 레코드 형식, 저장 데이터 항목의 표현 방법, 인덱스 유무, 내부 레코드의 물리적 순서를 나타냄. (하지만 블록이나 실린더를 이용한 물리적 저장 장치를 기술하는 의미는 아님)

(2) 논리 데이터저장소
① 논리 데이터 모델링과 관련하여 데이터 구조로 만들어진 데이터 저장소이다.
② 데이터베이스의 논리적 구성
- 개체(entity) : 표현하려는 유형, 무형 정보의 대상으로 존재하면서 서로 구별이 되는 것으로, 하나 이상의 속성으로 구성된다.
- 속성(attribute) : 개체의 특성이나 상태를 기술하는 것이다.(단독으로 존재하기는 어렵다.)
- 관계(relationship) : 개체간 또는 속성간의 상호 작용. (1 : 1, 1 : n, n : m)

2. 정규화

(1) 정규화의 개념

1) 개념

① 이상 문제를 해결하기 위해 어트리뷰트 간의 종속관계를 분석하여 여러개의 릴레이션으로 분해하는 과정.
② 릴레이션의 애트리뷰트, 엔티티, 관계성을 파악하여 데이터의 중복성을 최소화하는 과정
③ 논리적 설계 단계에서 수행
④ 정규화를 통해 릴레이션을 분해하면 일반적으로 연산시간이 증가한다.
⑤ 정규화 과정은 주어진 릴레이션 변수들의 모임을 더 바람직한 어떤 형태로 점차 유도해 가는 과정으로 특징지을 수 있다. 이 과정은 가역적(reversible)이다.

2) 목적

① 데이터베이스 연산의 여러 가지 이상을 없애기 위함이다.
② 데이터베이스의 물리적 구조나 물리적 처리에 영향을 주는 것이 아니라, 논리적 처리 및 품질에 큰 영향을 미친다.

3) 이상(anomaly)

- 어트리뷰트간에 존재하는 여러 종속관계를 하나의 릴레이션에 표현함으로 인해 발생하는 현상. (삽입이상, 삭제이상, 갱신이상)

[수강 릴레이션]

학번	**과목번호**	성적	학년
100	C413	A	4
100	E412	A	4
200	C123	B	3
300	C312	A	1
300	C324	C	1
300	C413	A	1
400	C312	A	4
400	C324	A	4
400	C413	B	4
400	E412	C	4
500	C312	B	2

① 삽입이상 : 원하지 않는 정보를 강제 삽입해야 하는 경우와 불필요한 데이터가 함께 삽입되는 경우이다.
ex) 위의 [수강 릴레이션]에서 만일 학번이 600인 학생이 2학년이라는 정보를 삽입하려고 할 때 교과목을 등록하지 않으면 삽입이 불가능하다.

② 삭제이상 : 튜플을 삭제함으로써 유지되어야 하는 정보까지도 연쇄 삭제(triggered delete)되는 정보의 손실(loss of information)을 삭제이상이라 한다.
ex) 위의 [수강 릴레이션]에서 만일 학번이 200인 학생이 과목 C123을 취소하여 이 튜플을 삭제할 경우 학년 3이라는 정보까지 함께 삭제된다.

③ 갱신이상 : 중복된 튜플 중에서 일부의 attribute만 갱신시킴으로써 정보의 모순성(inconsistency)이 생기는 현상이다.

ex) 위의 [수강 릴레이션]에서 만일 학번이 400인 학생의 학년을 4에서 3으로 변경하고자 할 때 모두 4번의 갱신이 필요하다.

4) 스키마 변환의 원리

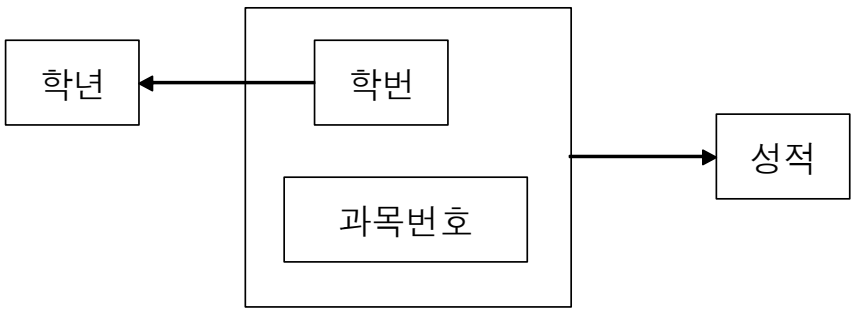
① 정보의 무손실 표현
② 데이터 중복성 감소
③ 분리의 원칙
④ 종속성 보존

5) 함수적 종속(functional dependency : FD)

① 어떤 릴레이션에서 속성들의 부분 집합을 X, Y라 할 때, 임의 튜플에서 X의 값이 Y의 값을 함수적으로 결정한다면, Y가 X에 함수적으로 종속되었다고 하고, 기호로는 X → Y 로 표기한다.

② 함수 종속 다이어그램(FD diagram)

[수강 릴레이션의 함수 종속 다이어그램]



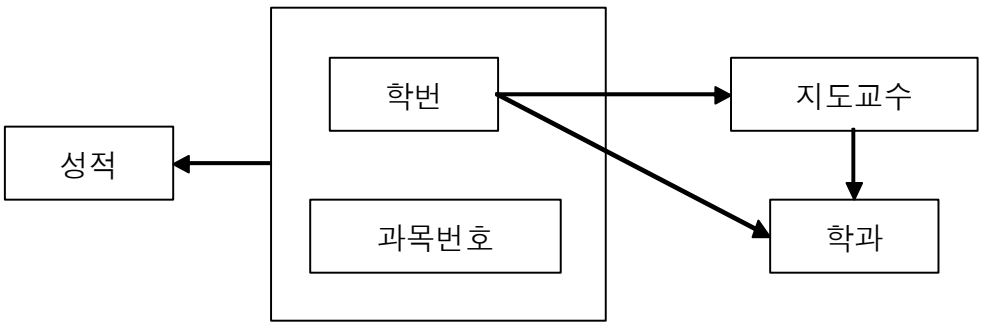
(2) 정규화 체계

학번	지도교수	학과	과목번호	성적
100	P1	컴퓨터	C413	A
100	P1	컴퓨터	E412	A
200	P2	전기	C123	B
300	P3	컴퓨터	C312	A
300	P3	컴퓨터	C324	C
300	P3	컴퓨터	C413	A
400	P1	컴퓨터	C312	A
400	P1	컴퓨터	C324	A
400	P1	컴퓨터	C413	B
400	P1	컴퓨터	E412	C

[수강지도 릴레이션]

수강지도 : (학번, 지도교수, 학과, 과목번호, 성적)
기본키 : (학번, 과목번호)
함수 종속 : (학번, 과목번호) → 성적
학번 → 지도교수
학번 → 학과
지도교수 → 학과

[수강지도 릴레이션의 함수 종속 다이어그램]



1) 제 1 정규형(INF)

어떤 릴레이션 R에 속한 모든 도메인이 원자값(atomic value)만으로 되어 있다면, 제 1 정규형(1NF)에 속한다.

① 모든 정규화 릴레이션은 제 1 정규형에 속한다.
② address와 같은 복합속성(composite attribute)은 원자적 도메인이 아니다.

㉠ 삽입 이상 : 어떤 학생이 교과목을 등록하지 않고는 그 학생의 지도교수를 삽입할 수가 없다. 즉, 학번이 500인 학생의 지도교수가 P4라는 사실을 삽입할 수 없다.

㉡ 삭제 이상 : 학번이 200인 학생이 과목 C123을 취소하여 이 튜플을 삭제할 경우 지도 교수가 P2라는 정보까지도 잃어버리게 된다.

㉢ 갱신 이상 : 학번이 400인 학생의 교수가 P1에서 P3으로 변경되었다면 학번 400이 나타난 모든 튜플을 P1에서 P3으로 변경해 주어야 한다.

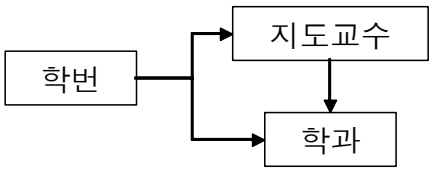
2) 제 2 정규형(2NF)

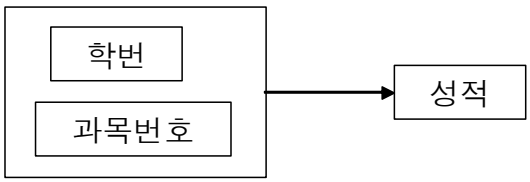
어떤 릴레이션 R이 1NF이고 키(기본)에 속하지 않은 애트리뷰트는 모두 기본키의 완전함수종속이면, 제 2 정규형(2NF)에 속한다.

① 1NF이면서 2NF가 아닌 릴레이션은 프로젝션을 하여 의미상으로 동등한 두 개의 2NF로 분해할 수 있고, 자연조인(natural join)을 통해 아무런 정보손실 없이 원래의 릴레이션으로 복귀가 가능하다.

② 2NF에서는 함수종속 관계 A→B, B→C이면 A→C가 성립하는 이행적 함수종속(transitive FD)이 존재한다. 이는 이상 현상의 원인이 된다.

지도 (학번, 지도교수, 학과)
학번 → 지도교수
학번 → 학과
지도교수 → 학과
수강 (학번, 과목번호, 성적)
(학번, 과목번호) → 성적





학번	지도교수	학과
100	P1	컴퓨터
200	P2	전기
300	P3	컴퓨터
400	P1	컴퓨터

[지도 릴레이션]

학번	과목번호	성적
100	C413	A
100	E412	A
200	C123	B
300	C312	A
300	C324	C
300	C413	A
400	C312	A
400	C324	A
400	C413	B
400	E412	C

[수강 릴레이션]

㉠ 삽입 이상 : 기본키 이외의 속성 삽입 시 기본키가 널(null)이 되는 문제가 있다. 즉, 어떤 교수가 특정 학과에 속하다는 사실을 삽입하려 할 때 이 교수의 지도를 받는 학생의 학번이 없이는 불가능하다.

㉡ 삭제 이상 : 학번이 200인 학생이 교수 P2와의 관계를 취소하여 튜플이 삭제되면 교수 P2가 어떤 학과에 속해 있다는 정보까지도 삭제된다.

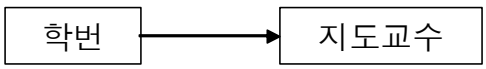
㉢ 갱신 이상 : 교수 P1의 학과가 컴퓨터에서 전기로 바뀐다면 학번 100과 400에 있는 학과의 값을 모두 변경해 주어야 한다.

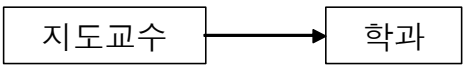
3) 제 3 정규형(3NF)

어떤 릴레이션 R이 2NF이고 키(기본)에 속하지 않은 모든 애트리뷰트들이 기본키에 이행적 함수종속이 아닐 때 제 3 정규형(3NF)에 속한다.

학생지도 (학번, 지도교수)
기본키 : {학번}, 외래키 : {지도교수} 참조 : 지도교수학과
학번 → 지도교수

지도교수학과 (지도교수, 학과)
기본키 : {지도교수}
지도교수 → 학과





학번	지도교수
100	P1
200	P2
300	P3
400	P1

[학생지도 릴레이션]

지도교수	학과
P1	컴퓨터
P2	전기
P3	컴퓨터

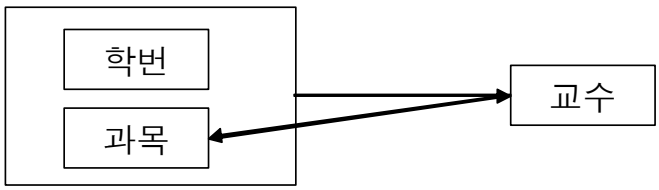
[지도교수학과 릴레이션]

4) 보이스/코드 정규형(BCNF)

릴레이션 R의 모든 결정자(determinant)가 후보키(candidate key)이면 릴레이션 R은 보이스/코드 정규형(BCNF)에 속한다.

① BCNF는 1NF, 2NF, 기본키, 이행 종속 등의 개념을 이용하지 않고 정의돌 수 있기 때문에 개념적으로 3NF보다 간단하다.

② BNCF는 제 3정규형보다 강력하다고 볼 수 있고 그래서 이 BCNF를 “강한 제 3 정규형(Strong 3NF)"이라고도 한다.



학번	과목	교수
100	프로그래밍	P1
100	자료구조	P2
200	프로그래밍	P1
200	자료구조	P3
300	자료구조	P3
300	프로그래밍	P4

[수강과목 릴레이션 (기본키 : {학번, 과목})]

㉠ 삽입 이상 : 교수 P5가 자료구조를 담당하게 되었다는 사실만 입력하고 싶을 때, 학생의 학번을 입력하지 않고는 삽입이 불가능하다.

㉡ 삭제 이상 : 학번이 100인 학생이 자료구조 과목을 취소하여 튜플이 삭제된다면 이때 교수 P2가 자료구조 과목을 담당하고 있다는 정보마저 없어져 버리게 된다.

㉢ 갱신 이상 : 교수 P1의 담당과목의 프로그래밍에서 자료구조로 변경되었다면 P1이 나타난 모든 튜플에 대해 갱신이 있어야 한다.

→ 이와 같은 변경 이상의 원인은 사실상 애트리뷰트 교수가 결정자이지만 후보키로 취급하고 있지 않기 때문이다.

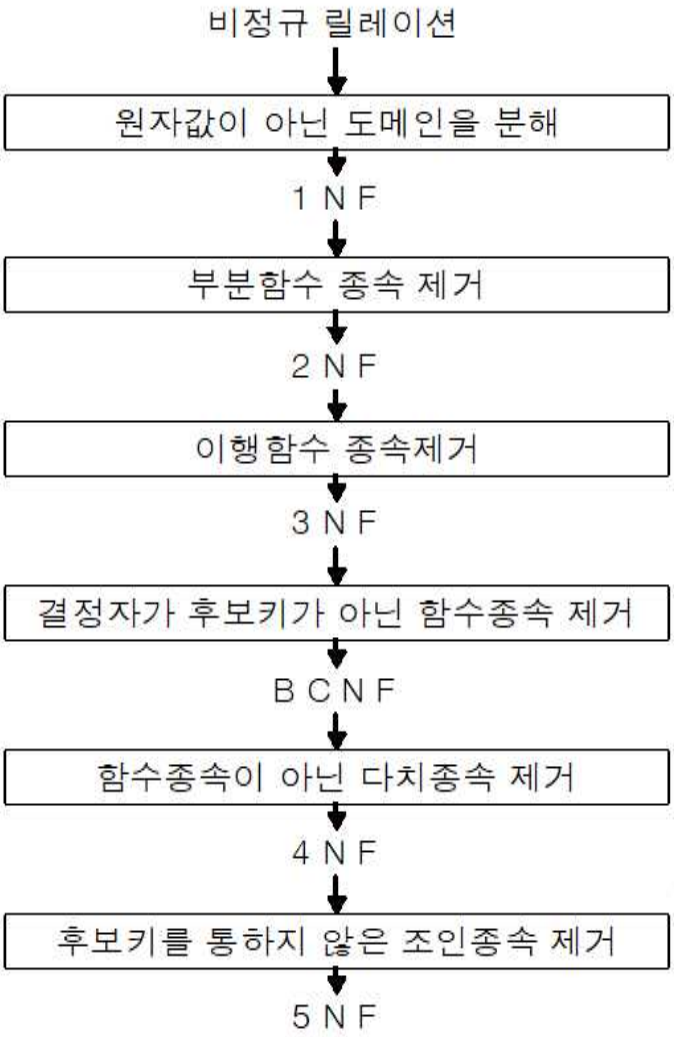
5) 제 4 정규형(4NF)

릴레이션 R에 MVD A↠B가 존재할 때 R의 모든 애트리뷰트들도 또한 A에 함수종속(즉, R의 모든 애트리뷰트 X에 대해 A→X이고 A가 후보키)이면 릴레이션 R은 제 4정규형에 속한다.

6) 제 5 정규형(5NF)

릴레이션 R에 존재하는 모든 조인 종속이 릴레이션 R의 후보키를 통해서만 성립된다면 릴레이션 R은 제 5 정규형에 속한다.

[정규화 과정]



3. 반정규화(De-Normalization)

① 반정규화의 정의

- 정규화되어있는 것을 정규화 이전 상태로 만드는 것을 말한다.
- 많은 조인에 의해 성능이 저하되거나 데이터 조회시 디스크 I/O량이 많을 때 부분적인 반정규화를 고려한다.

② 반정규화 절차

- 반정규화 대상조사
- 반정규화 대상을 다른 방법으로 처리 유도 할 수 있는지 검토
- 반정규화 적용

반정규화 대상조사	다른 방법 유도 검토	반정규화 적용
- 범위 처리 빈도수 조사 - 대량의 범위 처리 조사 - 통계성 프로세스 조사 - 테이블 조인 개수	- 뷰 테이블 - 클러스터링 적용 - 인덱스 조정 - 애플리케이션	- 테이블 반정규화 - 속성의 반정규화 - 관계의 반정규화

제3장 통합 구현

제1절 연계 데이터 구성하기

1. 통합 구현
통합 구현은 사용자들의 요구사항에 맞게 연계 시스템과 송신/수신 시스템 간의 관계를 적절히 구현하는 것이다.

① 송신 시스템
- 운영 데이터베이스에서 연계 데이터를 식별 및 추출하여 인터페이스 테이블(파일)로 생성하여 송신하는 시스템이다.
- 전송하고자 하는 데이터를 생성하여, 필요에 따라 변환 후 송신
- 송신 모듈, 송신 모니터링

② 수신 시스템
- 송신 시스템으로부터 수신한 테이블을 수신 시스템의 운영 데이터베이스나 환경에 맞게 변환하여 처리에 활용할 수 있도록 하는 시스템이다.
- 수신 받은 데이터를 정제, 가공하여 서비스
- 수신 모듈, 수신 모니터링

③ 연계 응용 시스템
- 송신시스템과 수신시스템을 연계해주는 서버나 시스템에 해당된다.
- 외부시시템간의 연계시에 적용되는 아키텍처
- 중계 모니터링, 변환 및 매핑

2. 연계 요구사항 분석
개발하고자 하는 응용소프트웨어와 관련된 외부 및 내부 모듈 간의 데이터 연계 요구사항을 분석할 수 있다.

- 사용자의 요구사항 분석은 연계 데이터와 연계 환경을 구성하기 위해 성능, 보안, 데이터 발생 유형 및 주기 등을 고려해야 한다.

3. 단위모듈 구현

(1) 공통모듈
- 전체 시스템 설계를 할 때에 각각의 서브 시스템에서 공통으로 사용되는 모듈들을 하나로 묶어서 놓은 소프트웨어 라이브러리를 말한다.
- 공통 모듈을 만드는 이유는 각각의 서브 시스템에서 제각각 모듈을 만들면 개발비도 중복으로 들어가고 표준화도 되지 않기 때문이다.
- 공통 모듈을 하나로 만들면 나중에 서브 시스템이 추가되더라도 공통 모듈은 재개발 없이 재사용이 가능하다는 장점이 있다.

(2) 단위모듈
- 화면 모듈, 화면에서 입력받은 데이터 처리를 위한 서비스 컴포넌트, 비즈니스 트랜잭션 컴포넌트 등이 있다.
- 공통모듈을 먼저 구현하고 이를 단위모듈 구현시에 재사용한다.

(3) 모듈화
① 모듈의 독립성이 높아야 모듈화가 잘 되었다고 평가할 수 있다.
② 결합도
- 결합도는 모듈들이 서로 관련되거나 연결된 정도를 나타낸다.
- 두 모듈 간의 상호 의존도
③ 응집도
- 한 모듈 내에 있는 처리 요소들 사이의 기능적인 연관 정도를 나타내며, 응집도가 높아야 좋은 모듈이 된다.

제2절 연계 매커니즘 구성하기

1. 연계 매커니즘
① 송신 시스템과 수신시스템으로 구성된다.
- 송신 시스템 : 데이터의 생성과 전송을 담당하는 시스템
- 수신 시스템 : 수신 및 운영 데이터베이스 반영을 담당하는 시스템
② 송신/수신 시스템의 현황을 모니터링하는 연계(중계)시스템을 설치 할 수 있다.
③ 연계 방식은 직접/간접 연계 방식으로 나눌 수 있다.

2. 직접연계방식
① 직접연계방식은 중간 매개체 없이 송신 시스템과 수신 시스템이 직접 연계되는 방식이다.
② 연계 및 구현이 단순하고 개발 소요 비용과 기간이 적게 소요된다.
③ 중간 매개체가 없기 때문에 데이터 연계처리 성능이 대체적으로 좋다.
④ 시스템간의 결합도가 높고, 시스템 변경에 민감하게 반응한다.
⑤ 보안을 위한 암복호화 처리와 비즈니스 로직 적용 등이 불가하다.

3. 간접연계방식
① 간접연계방식은 연계 솔루션과 같이 중간 매개체를 이용하여 연계하는 방식이다.
② 운영 데이터베이스에서 연계 데이터를 생성 및 변환과 송신 로그를 모니터링하는 구현 대상 솔루션에서 제공하는 송수신 엔진과 어댑터로 구성된다.
③ 중간 매개체가 존재하므로 서로 상이한 네트워크, 프로토콜 등 다양한 환경을 연계 및 통합 할 수 있다.
④ 시스템 간 인터페이스 변경 시에도 장애나 오류없이 서비스가 가능하며, 암복호화나 비즈니스 로직 적용이 용이하다.
⑤ 중간 매개체로 인한 성능이 저하될 수 있으며, 매커니즘이 복잡하다.

4. 연계 방식 분류
① 직접연계방식 : DB Link, DB API/Open API, JDBC
② 간접연계방식 : Web Service, Socket

※ 오픈 API
- Open Application Programmer Interface의 약자로, 인터넷 이용자가 웹 검색결과 및 사용자 인터페이스(UI)를 제공받는데 그치지 않고 직접 응용 프로그램과 서비스를 개발할 수 있도록 공개된 API.
- 산업 인력 관리 공단은 공공데이터인 자격정보, 기능경기, 직업방송 동영상 등 다양한 컨텐츠를 공공데이터포털(www.data.go.kr)에 공개하여 사용자가 원하는 컨텐츠로 재가공 할 수 있도록 제공중에 있으며, 큐넷은 자격정보(시험정보, 자격정보, 목록정보, 일정정보 등)를 API형태로 제공중에 있다

제3절 내외부 연계 모듈 구현하기

1. IDE 도구
(1) IDE의 개요
① 효율적으로 소프트웨어를 개발하기 위한 통합 개발 환경(IDE, Integrated Development Environment)이다.
② 기존의 소프트웨어 개발에서 코드 편집기, 디버거, 컴파일러, 인터프리터 등 분리되어 사용되던 것들을 통합하여 개발자에게 제공한다

(2) IDE의 종류
① 이클립스(Eclipse) : 자바, C, C++, PHP, JSP 언어 사용 가능.
② 라자루스(Lazarus) : 프리 파스칼, 파스칼 SDK 언어 사용 가능.
③ 엑스코드(X Code) : C, C++, 오브젝티브-C, 오브젝티브-C++, 자바, 애플스크립트, 코코아, Carbon, GNU 파스칼, 프리 파스칼, 에이다, C#, 펄, D, Swift 언어 사용 가능.
④ 비주얼 스튜디오(Visual Studio) : 비주얼 베이직, 비주얼 베이직 닷넷, 비주얼 C++, 비주얼 C 샤프, F 샤프 언어 사용 가능.
⑤ 제이빌더(J Builder) : JAVA 언어 사용 가능.
⑥ C++빌더(C++ Builder) : C, C++ 언어 사용 가능.

2. 협업도구

(1) 협업도구의 개요
① 소프트웨어 개발 프로젝트에 많은 개발자들이 참여하기 때문에 협업을 위한 도구가 필요하다.
② 프로젝트 임무를 수행하기 위해 각기 다른 장소에 있는 많은 사람들이 모여 IT 기술을 활용해 협력하고, 팀 단위의 활동을 수행하는 것을 가능하게하기 위해 협업도구가 필요하다.

(2) 협업도구의 기능
① 업무효율성 향상
② 정보 접근성 향상
③ 전체 이슈 진행 과정을 쉽게 파악
④ 직원관리(전자결재, 근태관리, 주소록)

(3) 협업도구의 종류
① 문서 공유 : 구글 드라이브
② 디자인 공유 : 레드 펜
③ 소스 공유 : 깃허브
④ 프로젝트 관리 : 트렐로, 레드마인, 지라
⑤ 일정 관리 : 구글 캘린더

3 형상관리 도구

(1) 소프트웨어 형상관리

1) 형상관리의 개요

① 형상(Configuration) : 소프트웨어 공학의 프로세스 부분으로부터 생성된 모든 정보항목의 집합체
② 소프트웨어 형상관리 항목(SCI, Software Configuration Item)

㉠ 분석서
㉡ 설계서
㉢ 프로그램(원시코드, 목적코드, 명령어 파일, 자료 파일, 테스트 파일)
㉣ 사용자 지침서

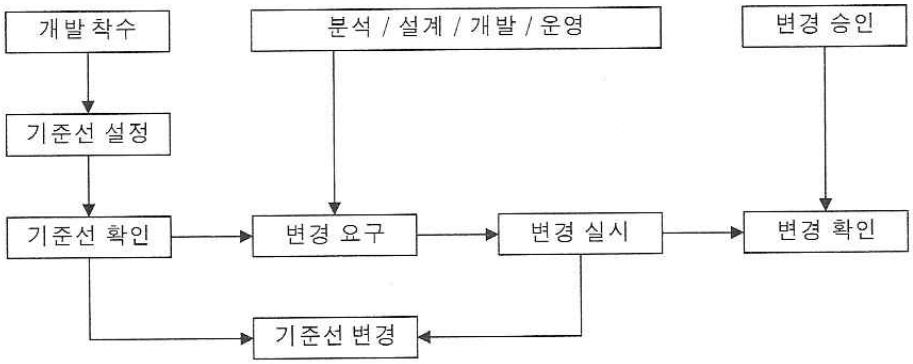
③ 형상관리(SCM, Software Configuration Management)

㉠ 소프트웨어에 대한 변경을 철저히 관리하기 위해 개발된 일련의 활동
㉡ 소프트웨어를 이루는 부품의 Baseline(변경통제 시점)을 정하고 변경을 철저히 통제하는 것

④ 베이스라인(Baseline)

㉠ 정식으로 검토되고 합의된 명세서나 제품으로서, 이것으로부터 앞으로의 개발을 위한 바탕 역할을 하며, 정식 변경 통제 절차들을 통해서만 변경될 수 있는 것(IEEE)
㉡ 정당화될 수 있는 변경에 심하게 저항하지 않으면서 변경을 통제하게 도와주는 하나의 소프트웨어 형상관리 개념이다.

⑤ 전체 소프트웨어 프로세스에 적용되는 “보호활동” 이다.



[형상관리 상태도]

2) 형상관리를 위한 조직(형상관리 위원회(팀))
① 분석가 : 사용자와 상의하여 무엇이 문제이며 어떤 기능향상 및 개작이 필요한가를 결정한다.
② 프로그래머 : 분석가와 협동하여 문제의 원인을 찾아내고 변경의 형태와 내용을 알아낸다. 실제 프로그램의 수정을 담당한다.
③ 프로그램 사서 : 문서와 코드에 대한 변경을 계속 보관하고 관리한다.

3) 형상관리의 목적
① 가시성의 결여(Lack of Visibility)에 대한 문제 해결
② 통제의 어려움(Lack of Control)에 대한 문제 해결
③ 추적의 결여(Lack of Traceability)에 대한 문제 해결
④ 감시의 미비(Lack of Monitoring)에 대한 문제 해결
⑤ 무절제한 변경(Incontrolled Change)에 대한 문제 해결

(2) 형상관리의 기능

1) 형상 식별(Identification)
① 형상 식별은 소프트웨어 형상의 모든 항목에 대해 의미있고 항구적인 명명을 보증하는 소프트웨어 형상관리 활동이다.
② 형상관리 항목(SCI, Software Configuration Item)에 대해 관리 목록 번호를 부여하고 나무 구조를 표현하여 저장한다. 이는 관련 문서에 대한 추적을 용이하게 한다.
③ 통제가 쉽도록 “누가, 언제, 무엇을 왜 정의하였는가?” 하는 정보를 생성하며, 기준선을 설정한다.

2) 형상 통제(Control)
① 식별된 SCI의 변경요구를 검토하고 승인하여 현재의 베이스라인에 적절히 반영될 수 있도록 통제하기 위한 형상관리 활동이다.
② 변경 요구(Change Request)의 제기 → 변경 요청서(Change Report) 작성(변경 요청서는 CCA(Change Control Authority)에 의해 변경의 상태나 우선순위 등 최종 결정을 내리도록 사용자 또는 프로그래머에 의해 작성) → 공학 변경 명령ECO(Engineering Change Order)
③ 형상통제는 소프트웨어 유지보수를 위한 변경관리와 일치한다.

3) 형상 감사(Auditing)
① 변경이 적절하게 시행되었는지 객관적인 검증과 확인(V&V) 과정을 거쳐 새로운 형상의 무결성을 확

보하기 위한 활동이다.

① 정형 검토 회의(Formal Technical Review)
- 수정 완료된 형상 객체의 기술적인 정확성에 초점을 둔다.
- 검토자들은 SCI를 산정하여 다른 SCI와의 일관 혹은 잠재적인 부작용 유무를 검토한다.

② 소프트웨어 형상 감사(Software Configuration Audit)
- 검토시 일반적으로 고려되지 않은 특성들에 대해 형상 객체를 산정함으로써 FTR을 보완한다.
- SCM이 정식 활동일 경우에는 품질보증 조직과 별도로 SCM 감사를 실시한다.

4) 형상 보고(Status Accounting)
① 형상 식별, 변경 통제, 형상 감사 기능의 수행 결과를 기록하고 데이터베이스에 의해 관리를 하며 이에 대한 보고서를 작성하는 활동이다.
② 형상 상태 보고(CSR, Configuration Status Reporting)라고도 한다.

(3) 형상관리 도구

1) 형상관리 도구의 개요
① 프로그램 소스를 특정 저장소에 저장해둔 것을 내려 받아 수정 후 업로드시키고 다른 개발자가 개발한 최신 소스를 내려받아 분석 및 빌드하도록 도와주는 도구이다.
② 형상관리는 일반적으로 버전 관리 (version control, revision control), 소스 관리 (source control), 소스 코드 관리 (source code management, SCM)와 동일한 의미로 사용한다.
③ 소스코드 버전 관리 툴의 종류로는 CVS, SVN, GIT,,, 등이 있다.

2) 형상관리 도구의 주요 기능
① 소프트웨어 프로젝트를 빌드하기 위한 소스코드, 이미지, 스크립트 등의 저장소
② 이러한 파일들의 변경을 체계적으로 관리, 제어(변경기록 추적, 특정 시점 파일 상태 조회)
③ 팀내 다수의 개발자와 협업(작성된 소스 코드와 변경사항을 확인 및 수정)을 위한 도구와 매커니즘)
④ 장애 혹은 기능상 필요할 때 이전 버전으로 소프트웨어를 원상복구할 수 있다.
⑤ 동일한 소프트웨어를 여러 개의 버전으로 분기해서 개발할 필요가 잇는 경우에 유용

3) 형상관리 도구의 구성 요소

구분	설명
Repository	• 프로젝트의 프로그램 소스를 포함한 형상항목이 저장되는 장소 • 소스뿐만 아니라 소스의 변경사항도 모두 저장 • 네트워크를 통해서 여러 사람이 접근 가능함
checkout	• 저장소에서 소스 및 버전관리 파일들을 받아 옴
commit	• 소스를 수정 및 삭제, 새파일 추가등의 변경사항을 저장소에 갱신
Update	• 체크아웃을 통해서 소스를 가져왔다 하더라도 다른 사람이 커밋을 하면 로컬 소스코드가 달라지는데 이때, update 명령어를 통해서 저장소에 있는 최신 버전의 소스를 가져올 수 있다. • 로컬 소스 코드와 저장소에 있는 소스 코드를 비교하여 차이가 발생하는 부분만 바꿈

4) 형상관리 도구의 종류

① GitHub

구분	설명
개념	• 분산형 버전관리 시스템(2005년 리누스 토발즈와 주니오 하마노가 개발) • 개발자가 중앙 서버에 접속하지 않아도 코딩 가능 • 어떤 코드를 누가 수정했는지 기록, 추적 가능
특징	<<장점>> • Repository의 완전한 복사본을 로컬에 저장할 수 있다. • 안정적이고, 속도가 빠름(svn, cvs 보다 우수) • Branch merge를 할 경우 리비전을 지정하지 않아도 되므로 편리하다.(해당 branch가 언제 생겨났는지 자동적으로 파악된다.) • 원격 레파지토리 장애에도 문제없이 버전관리가 가능함 (로컬에 저장하기 때문에 장소와 시간에 구애 받지 않고 협업 가능) • SVN과 다르게 Commit은 로컬 저장소에 저장되고 Push를 통해 원격 저장소에 저장됨. • 서버에서 소스를 수신 받을시 Pull 기능을 사용함. <<단점>> • 다른 툴에 비해 다소 사용하기 무겁고, 첫 사용시 어렵다. (개별 로컬 파일을 가질 수 있음)

② SVN

구분	설명
개념	• SubVersion의 줄임말(CSV의 단점 보안(2000년)) • 파일과 디렉토리의 삭제, 이동, 이름 변경, 복사 등을 지원(버전 관리 O) • Trunk, Branches, Tags 로 구성 • import, commit, checkout, revert, switch, update, merge 등의 명령어
특징	<<장점>> • 효율적인 Branch및 Merge기능과 작업의 무결성을 보장 • 원자적 커밋을 통해 다른 사용자의 커밋과 엉키지 않으며, 커밋 실패 시 롤백 기능을 지원한다. • 파일과 디렉토리의 삭제, 이동, 이름 변경, 복사 등을 지원한다. • 소스파일 이외에 이진파일도 효율적으로 저장할 수 있다. • 디렉토리에 대해 버전 관리를 할 수 있다. (디렉토리 전체를 이동시키거나 복사가 가능/리비전 기록 유지) • 처리 속도가 상대적으로 빠르다. <<단점>> • 자동 생성되는 .svn 디렉토리로 인해 저장소가 다소 지저분한 느낌을 준다. • 잦은 커밋으로 인해 리비전 번호가 크게 증가할 수 있다. • 개별 개발자만의 개발 이력을 가질 수 없다.

③ CVS

구분	설명
개념	• 개발과정에서 사용하는 파일들의 변경내역을 관리
특징	<<장점>> • 하나의 파일에 대한 동시작업이 가능 • Merge, Branch, Tag, Compare 기능 지원 • Unix, Linux, Windows 등 다양한 운영체제 지원 • 레파지토리를 백업하는 것만으로도 프로젝트의 백업이 됨 <<단점>> • CVS 저장소의 파일들은 이름 변경이 불가하므로, 제거한 뒤 다시 추가해야 한다. • 아스키 코드를 지원하며, 유니코드는 제한적으로 지원한다. • 속도가 상대적으로 느리다. • 커밋 실패 시 롤백이 지원되지 않는다.

제4장 서버프로그램 구현

제1절 모듈화

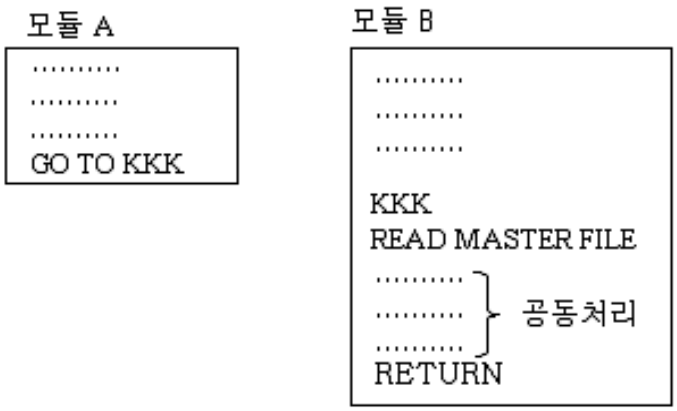
- 상호독립된 모듈은 기능단위로 잘 분해되고 접속관계가 단순하여 개발이 용이하며, 유지보수시 수정에 따른 파급효과를 최소화할 수 있다.
- 모듈간의 결합도 최소화, 응집도 최대화

1) 결합도

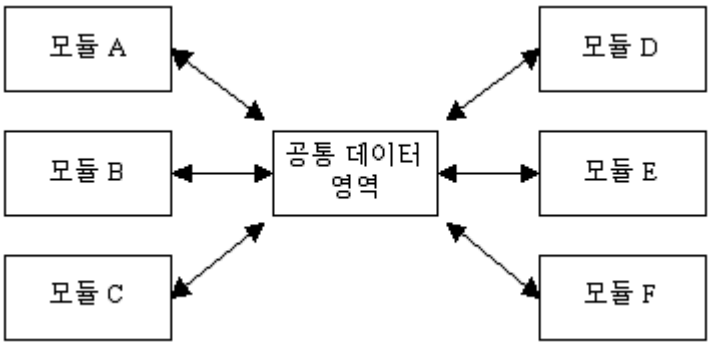
- 결합도는 모듈들이 서로 관련되거나 연결된 정도를 나타낸다.
- 두 모듈 간의 상호 의존도
- 낮은 결합도를 유지해야 바람직하다.

1. 내용 결합도(content coupling)	결합도가 높음
2. 공통 결합도(common coupling)	
3. 외부 결합도(external coupling)	
4. 제어 결합도(control coupling)	
5. 스탬프 결합도(stamp coupling)	
6. 자료 결합도(data coupling)	결합도가 낮음

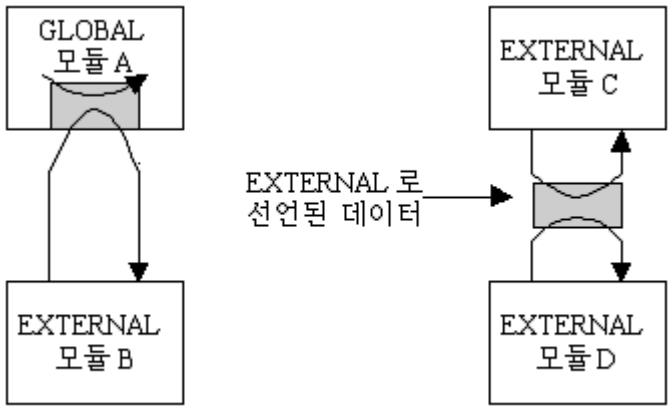
① 내용 결합도 : 어떤 모듈을 호출하여 사용하고자 할 경우에 그 모듈의 내용을 미리 조사하여 알고 있지 않으면 사용할 수가 없는 경우에는 이들 모듈이 내용적으로 결합되어 있기 때문이며, 이를 내용 결합도라고 한다.



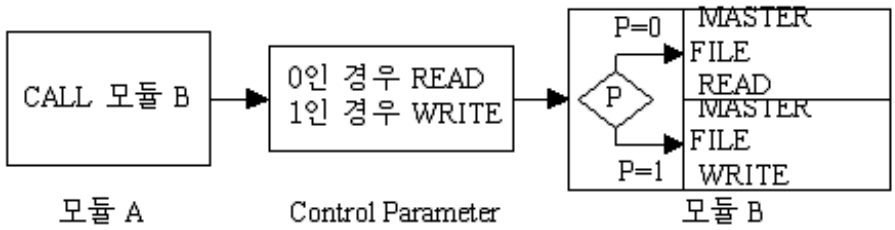
② 공통 결합도 : 공통 결합도는 하나의 기억 장소에 공동의 자료 영역을 설정한 후, 한 모듈이 그 기억 장소에 자료를 전송하면 다른 모듈은 기억 장소를 조회함으로써 정보를 전달받는 방식을 취할 때 발생된다.



③ 외부 결합도 : 일련의 모듈들이 동일한 광역 데이터 아이템(단일 필드 변수)을 사용하면 외부 결합도가 된다.



④ 제어 결합도 : 어떤 모듈이 다른 모듈을 호출할 경우, 제어 정보를 파라미터로 넘겨주는 경우 이들 두 모듈은 제어 결합도를 가졌다고 한다.



⑤ 스탬프 결합도 : 한 그룹의 모듈들이 동일한 비광역 데이터 구조를 사용한다면 스탬프 결합도가 될 수 있다. 예로서, 모듈 A가 모듈 B를 호출하여 종업원 개인 레코드를 전송하고 A와 B가 둘 다 그 레코드의 형태나 구조에 영향을 받기 쉽다면, A와 B는 스탬프 결합도를 가진 것이다. 스탬프 결합도는 모듈들간의 불필요한 연관 관계를 형성하므로 가능한 한 회피하는 것이 좋다.

⑥ 자료 결합도 : 모듈간의 결합도 중 가장 바람직한 결합도는 자료 결합도이다.

2) 응집도

- 한 모듈 내에 있는 처리 요소들 사이의 기능적인 연관 정도를 나타내며, 응집도가 높아야 좋은 모듈이 된다.
- 한 모듈 내에 필요한 함수와 데이터들의 친화력을 측정하는데 사용

1. 우연적 응집도(coincidental cohesion)	응집도가 낮음
2. 논리적 응집도(logical cohesion)	
3. 시간적 응집도(temporal cohesion)	
4. 절차적 응집도(procedural cohesion)	
5. 통신적 응집도(communicational cohesion)	
6. 순차적 응집도(sequential cohesion)	
7. 기능적 응집도(functional cohesion)	응집도가 높음

① 우연적 응집도 : 우연적 응집도는 모듈 내부의 각 요소들이 서로 관계없는 것들이 모인 경우로 응집력이 가장 낮다.

② 논리적 응집도 : 논리적으로 서로 관련이 있는 요소를 모아 하나의 모듈로 한 경우, 그 모듈의 기능은 이 모듈을 참조할 때 어떤 파라미터를 주느냐에 따라 다르게 된다.

③ 시간적 응집도 : 어느 특정한 시간에 처리되는 몇 개의 기능을 모아 한 모듈로 한 경우, 이들 기능은 시간적인 관계로 결속되는 경우가 된다. 예를 들어, 프로그램의 초기화 모듈이나 프로그램 종료 모듈이 이에 해당된다.

④ 절차적 응집도 : 어떤 모듈이 다음 조건을 충족시킬 때 절차적 응집도를 가진다고 한다.
- 다수의 관련 기능을 수행
- 기능들을 순차적으로 수행

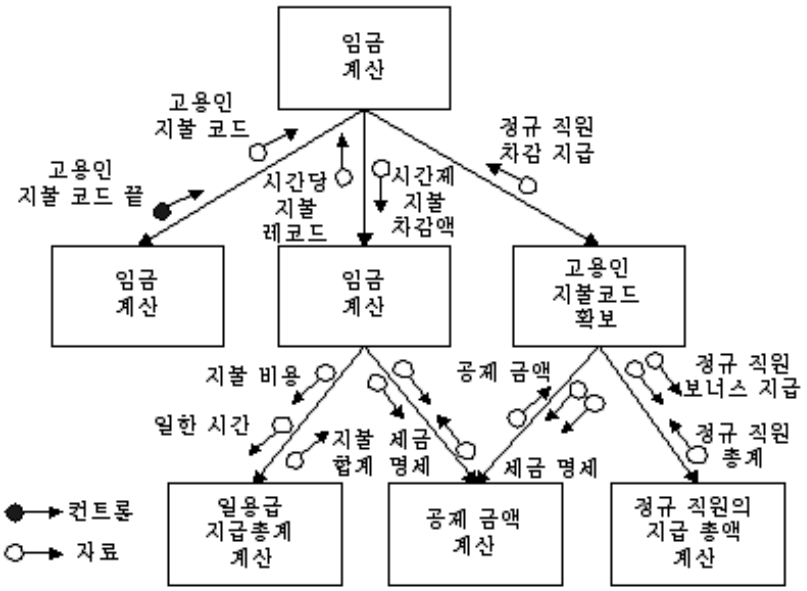
⑤ 통신적 응집도 : 단말기로부터 데이터를 읽고 검사하여 데이터베이스에 입력하는 모듈은 그 안의 기능들이 그 레코드의 사용과 연관되므로 통신적 응집도를 갖는다.

⑥ 순차적 응집도 : 순차적 응집도는 실행되는 순서가 서로 밀접한 관계를 갖는 기능을 모아 한 모듈로 구성한 것으로 흔히 어떤 프로그램을 작성할 때 순서도를 작성하는데 이 경우에는 순차적 응집도를 갖는 모듈이 되기 쉽다.

⑦ 기능적 응집도 : 모듈내의 모든 요소가 한 가지 기능을 수행하기 위해 구성될 때, 이들 요소는 기능적 응집도로 결속되어 있다고 한다.

제2절 구조도(Structure Chart)

① 시스템 기능을 몇 개의 기능으로 분할하여 모듈로 나타내고, 모듈간의 인터페이스를 계층 구조로 표현한 도형이다.
② 구조도에서 사각형은 모듈, 백색원의 화살표는 매개변수를 이용한 자료의 이동, 흑색원의 화살표는 실행의 방향을 나타내는 제어흐름, 마름모는 선택, 곡선 화살표는 반복을 나타낸다.
③ 의사결정 박스(선택)를 갖지 않는다는 점과 각 작업에 대한 순서를 표시하지 않는다는 점에서 순서도와 차이가 있다. 하지만 구조도는 일반적으로 보통 위에서 아래로, 왼쪽에서 오른쪽으로의 순서로 실행된다.



[구조도의 예]

④ 구조도는 순차, 선택, 반복의 제어구조를 나타낼 수 있다.
⑤ 팬 입력은 특정 모듈을 직접 제어하는 모듈의 수이다.
⑥ 팬 출력은 한 모듈에 의해 직접 제어되는 모듈의 수이다.

제5장 인터페이스 구현

제1절 인터페이스 설계 확인

1. 인터페이스 기능확인

(1) 인터페이스
- 인터페이스 기능은 내부나 외부이 모듈간의 연계 기능을 말한다.
- 자바언어에서는 인터페이스를 다음과 같이 정의하기도 한다. 일반 메서드 또는 멤버 변수를 가질 수 없고, 오직 추상 메서드와 상수만을 멤버로 가질 수 있다.

(2) 인터페이스 설계서
- 시스템 내·외부 인터페이스를 식별하고 인터페이스의 명세를 기술한다.
- 인터페이스 명세서는 각각의 인터페이스 설계를 상세하게 적어 놓은 문서이다.

[인터페이스 목록 항목]

구분		설명
송신	인터페이스번호	송신 시스템의 인터페이스 일련번호를 기입한다
	일련번호	한 개 송신단위에서 여러 개의 서브시스템으로 동시에 전송되는 경우에는 순차적으로 기술한다.
	시스템명	송신 시스템이름을 기술한다
	프로그램 ID	송신에 해당하는 프로그램 ID를 기입한다.
전달	처리형태	인터페이스를 처리하는 형태를 기술한다. Batch / Online 등
	인터페이스방식	통신 프로토콜 및 통신 기술 방식을 기술한다.
	발생빈도	인터페이스 발생빈도를 기술한다. "회수/주기"의 형식으로 기술한다.
수신	상대 담당자	수신 시스템의 업무담당자명을 기술한다.
	프로그램 ID	수신과 관련된 프로그램 ID를 기입한다.
	수신 시스템명	인터페이스 수신 시스템명을 기술한다.
	일련번호	수신 시스템의 동일 인터페이스가 여러 시스템에서 동시에 수신을 받는 경우에 순차적으로 번호를 부여한다.
	수신번호	수신 식별번호를 기입한다
관련 요구사항 ID		해당 인터페이스와 관련된 분석단계의 “사용자 요구사항 정의서”의 요구사항ID를 기입한다.
비고		특이사항 등을 기입한다.

2. 데이터 표준 확인

(1) 인터페이스 데이터 표준
- 이질적인 시스템간에 의사소통(송·수신)시 data 형식이 맞지 않는 경우가 발생한다. 이때, data 연계 코드 변환 및 매핑 처리가 필요하다.
- 송·수신되는 연계 정보에 포함된 코드를 변환하는 방법에는 송신 시스템 코드를 수신 시스템 코드로 매핑해주는 방법과 송·수신 시스템에서 사용되는 코드를 통합하여 표준화한 후 매핑해주는 방법이 있다.

(2) EAI(Enterprise Application Integration, 기업 내외부 정보시스템 통합)
- 기업의 내부 및 외부 애플리케이션 사이의 통합을 위해 제공되는 프로세스, 기술 및 툴의 집합

1) EAI 구성요소

구성요소	설명
EAI Platform	- 데이터 전송을 보장하는 메시지 큐와 트랜잭션 미들웨어 기능 수행 - 유연성이 있고, 대규모 사용자 환경까지 사용할 수 있는 확장성 보장
Application Adaptor	- 다양한 패키지 어플리케이션 및 기업에서 자체적으로 개발한 어플리케이션을 신속하고 재사용성이 높은 인터페이스 지원 - DB, CRM, ERP, DW 등 어플리케이션을 연결하는 어댑터
브로커(Broker)	- 시스템 상호간 데이터가 전송될 때, 데이터 포맷과 코드를 변환하는 솔루션 - 일종의 Mediator &Wrapper 기능 수행
Business Workflow	- 미리 정의된 기업의 비즈니스 workflow에 따라 업무 처리해주는 기능

2) 주요 기능
① 비지니스 프로세스 관리기능
- 각 업무 시스템 및 app 상호간에 데이터의 교환과 더불어 각 업무에 대한 흐름을 어떤 시점 또는 어떤 이벤트에 따라서 어디에서 어디로 업무가 진행되어야 하는지를 정의하고 운용할 수 있는 기능
② 데이터 브로커 기능
- App 상호간에 중개되는 데이터를 자동변환하여 전달하고 데이터 소스에서 지정된 대상 시스템까지의 연결
③ APP 접근 기능
- 패키지 app 또는 메인프레임과 같은 이기종 시스템과의 접속을 위한 기능
- 해당 SW와 플랫폼 사이에 위치하며 데이터 중개 및 app연동의 인터페이스를 제공
④ 데이터접근 기능
- 데이터에 대한 통합을 담당하는 영역으로 주로 데이터의 전송, 타입 변환, 데이터의 정제 및 추출 기능

⑤ 플랫폼 기능
- EAI의 기반이 되는 app서버 또는 미들웨어로 구성되어 있는 영역으로 EAI를 안정성있게 실행하고 EAI 모든 기능들이 정상적으로 동작할 수 있도록 하는 기능

3) EAI 유형

구분	설명
Point- to - Point	- 1:1 방식으로 애플리케이션 통합 수행
Hub & Spoke	- 모든 데이터가 허브를 통해 전송 - 데이터 전송이 보장되며, 유지보수 비용 절감
메세징 버스	- 데이터 전송하는데 버스를 이용함으로 병목 현상 발생가능 - 대량의 데이터 교환에 적합
하이브리드	- Hub & spoke방식과, 메시징 버스 방식의 통합 - 유연한 통합 작업 가능

4) EAI 통합 4단계
① 데이터 : 데이타 추출, 데이터 변환, 데이터 라우팅 및 갱신
② 애플리케이션 : 메시지, API 통한 직접적 수행
③ 비즈니스 로직 : 분산 비즈니스 오브젝트를 통한 시스템별 비즈니스 로직 프로비저닝
④ 사용자 인터페이스 : 애플리케이션의 입출력 포인트, 전용시스템에 유용

제2절 인터페이스 기능 구현

- 인터페이스를 구현하는 대표적인 방법으로는 데이터 통신을 이용한 인터페이스 구현 방법과 인터페이스 테이블을 이용한 인터페이스 구현 방법으로 나눌 수 있다.
- 데이터 통신을 사용한 인터페이스에서 예외 처리 방법은 AJAX 방식을 사용하여 JSON 객체를 전달하므로 AJAX 방식의 예외 처리 방식에 따라 JSON 객체 인터페이스 송수신 시 구현한다.

1. 인터페이스 구현을 위한 도구

(1) JSON(JavaScript Object Notation)

① 속성-값 쌍(attribute-value pairs and array data types (or any other serializable value)) 또는 "키-값 쌍"으로 이루어진 데이터 오브젝트를 전달하기 위해 인간이 읽을 수 있는 텍스트를 사용하는 개방형 표준 형식이다.
② 비동기 브라우저/서버 통신 (AJAX)을 위해 넓게는 XML(AJAX가 사용)을 대체하는 주요 데이터 포맷이다.
③ JSON은 특히 인터넷에서 자료를 주고 받을 때 그 자료를 표현하는 방법으로 알려져 있다.
④ 웹과 컴퓨터 프로그램에서 용량이 적은 데이터를 교환하기 위해 데이터 객체를 속성/값의 쌍 형태로 표현하는 형식으로 자바스크립트를 토대로 개발되어진 형식이다.
⑤ 자료의 종류에 큰 제한은 없으며, 특히 컴퓨터 프로그램의 변수값을 표현하는 데 적합하다.

```
- 형식 : { String key : String value }
- 예
  {
          "firstName" : "Kim",
          "lastName" : "GilDong",
          "age" : 25,
          "email" : "aaa@aaaaa.com"
  }
```

(2) XML(eXtensible Markup Language)

① HTML의 단점을 보완한 인터넷 언어로, SGML의 복잡한 단점을 개선한 다목적 마크업 언어이다.

② 웹상에서 구조화된 문서를 상호교환 가능하도록 설계된 웹 표준 문서 포맷으로 메타 데이터 정의가 명확하다.

③ 사용자가 새로운 태그와 속성을 정의할 수 있는 확장성을 가진다.

④ 유니코드를 사용하여 전 세계의 모든 문자를 처리 가능하며 장치와 시스템에 독립적이다.

2. AJAX(asynchronous JavaScript and XML)

① JavaScript를 사용한 비동기 통신 기술이다.

② 브라우저가 갖고 있는 XMLHttpRequest 객체를 이용해서 전체 페이지를 새로 고치지 않고 페이지의 일부만을 위한 데이터를 로드하는 기법이며 JavaScript를 사용한 비동기 통신, 클라이언트와 서버 간에 XML 데이터를 주고받는 기술이라 할 수 있다.

③ Ajax 경우 HTML 페이지 전체가 아닌 일부분만 갱신할 수 있도록 하게 해준다.

④ JSON 이나 XML 형태로 필요한 데이터만 받아 갱신하기 때문에 효율성이 좋다.

3절 인터페이스 구현 검증

1. 설계 산출물

- 인터페이스 구현 검증에 사용되는 산출물은 인터페이스 (정의서)명세서와 인터페이스 구현 검증 시나리오를 기반으로 테스트 자동화 수행 도구를 사용하여 테스트 검증을 진행한다.
- 인터페이스 구현 검증에 필요한 설계 산출물은 인터페이스 명세서와 인터페이스 단위 및 통합 테스트 설계서이다.

2. 인터페이스 명세서

- 인터페이스 정의서에는 송신 시스템과 수신 시스템 간의 인터페이스 현황을 작성한다
- 인터페이스 명세서는 인터페이스 정의서에 작성한 인터페이스 ID 별로 송수신하는 데이터 타입, 길이 등 인터페이스 항목을 상세히 작성한다.(인터페이스번호, 송신시스템(시스템명, 데이터저장소명, 속성명, 데이터타입, 길이), 송신프로그램 ID, 수신시스템(데이터저장소명, 속성명, 데이터타입, 길이, 시스템명), 수신프로그램 ID)

[인터페이스 구현 검증 도구]

제품명	세부정보
xUnit	java(Junit), C++(Cppunit), .Net(Nunit) 등 다양한 언어를 지원하는 단위 테스트 프레임워크
STAF	서비스 호출, 컴포넌트 재사용 등 다양한 환경을 지원하는 테스트 프레임워크
FitNesse	웹기반 테스트케이스 설계/실행/결과확인 등을 지원하는 테스트 프레임워크
NTAF	NHN 테스트 자동화 프레임워크이며, STAF와 FitNesse를 통합
Selenium	다양한 브라우저 지원 및 개발언어를 지원하는 웹애플리케이션 테스트 프레임워크
watir	Ruby 기반 웹애플리케이션 테스트 프레임워크

제6장 화면 설계

제1절 UI 요구사항 확인

1. UI 표준

(1) UI(사용자 인터페이스)의 개념
① 외부설계의 한 종류이며, 소프트웨어와 조직 환경과의 인터페이스를 설계하는 과정이다.

② 사용자 인터페이스 평가 기준
㉠ 배우기 쉬움 : 소프트웨어를 사용할 수 있게 되기까지 배우는 데 걸리는 시간
㉡ 속도 : 특정 기능을 수행시키는 데 걸리는 시간
㉢ 사용 중 오류의 빈도 : 원하는 작업을 수행시킬 때 사용자가 범한 오류의 빈도
㉣ 사용자의 만족 : 시스템에 대한 사용자의 반응
㉤ 사용법의 유지 : 시스템 사용에 대한 지식이 얼마나 쉽게 기억될 수 있는가?

(2) 사용자 인터페이스론 규칙
- 일관성을 유지할 것
- 시작, 중간, 종료가 분명하도록 설계할 것
- 오류 처리 기능 간단히 할 것
- 단순화시켜 기억의 필요성을 줄일 것
- 단축키를 제공할 것

(3) UI 종류
- CUI(Character based UI): 문자 방식의 명령어 입력 사용자 인터페이스
- GUI(Graphic UI): 그래픽 환경 기반의 마우스 입력 사용자 인터페이스
- NUI(Natural UI): 사용자의 말과 행동 기반의 제스쳐 입력 인터페이스
- 기타(웹 기반 인터페이스(웹 사용자 인터페이스, WUI), 터치 사용자 인터페이스, 텍스트 사용자 인터페이스(TUI)...)

(4) UI 표준
- 전체 시스템에 공통으로 적용되는 화면 간 이동, 화면 구성 등에 대한 규약을 UI 표준 혹은 표준 UI라고 한다.
- 대표적인 UI 표준은 MS Vista UX Guideline, Apple OS X UX Guideline 이 있다.

2. UI 지침

(1) UI 기본원칙
① 직관성(Intuitiveness)
- 누구나 쉽게 이해하고 사용할 수 있도록 제작
② 유효성(Efficiency)
- 정확하고 완벽하게 사용자의 목표가 달성될 수 있도록 제작
③ 학습성(Learnability)
- 초보와 숙련자 모두가 쉽게 배우고 사용할 수 있게 제작
④ 유연성(Flexibility)
- 사용자의 인터랙션을 최대한 포용하고, 실수를 방지할 수 있도록 제작

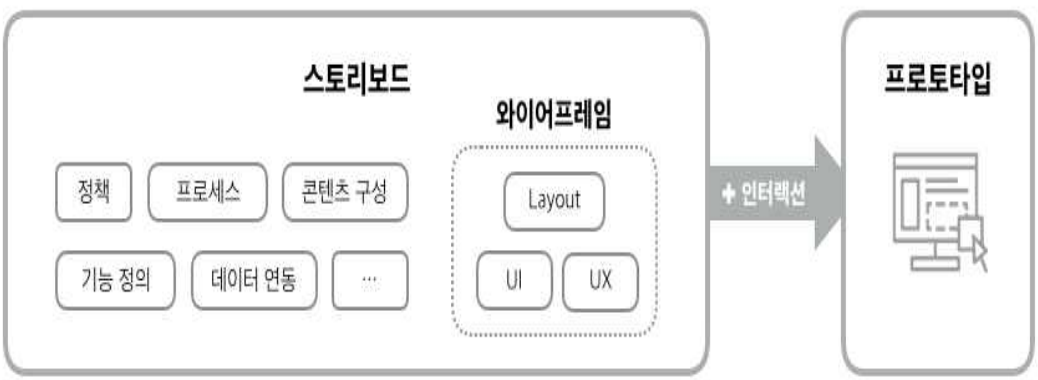
3. 스토리보드
- 기획단계에서 이루어지고 이후에 개발될 모형타입을 만드는 것을 일컫는다.

(1) 스토리보드(Storyboard)
- 디자이너/개발자가 참고하는 최종적인 산출문서로써 정책, 비즈니스, 프로세스, 콘텐츠, 구성, 와이어프레임, 기능 정의, 데이터베이스 연동 등 서비스 구축을 위한 모든 정보가 담겨있는 문서이다. 현업에서 해당 문서를 바탕으로 커뮤니케이션을 진행한다. 스토리보드의 툴로는 파워포인트, 키노트, 스케치 등이 있다.

(2) 와이어프레임(Wireframe)
- 와이어프레임은 화면 단위의 레이아웃을 설계하는 작업이다. 의사소통 관계자들과 레이아웃을 협의하거나 서비스의 간략한 흐름을 공유하기 위해 사용하며, UI, UX 설계에 집중되어 있다. 와이어프레임 도구는 손그림, 파워포인트, 스케치, 일러스트 그리고 포토샵이 있다.

(3) 프로토타입(Prototype)
- 프로토타입은 실제 서비스와 흡사한 모형을 만드는 작업이다. 정적인 화면으로 설계된 와이어프레임 또는 스토리보드에 인터랙션(동적효과)를 적용함으로써, 실제 구현된 것처럼 시뮬레이션 할 수 있으며 단시간에 구현이 가능하기 때문에 사용자 경험에 대한 테스터를 진행해 볼 수 있다. 이를 통해 설계 단계의 리스크를 사전에 예방할 수 있다.

(4) 와이어프레임/스토리보드/프로토타입 관계도



본

제2절 UI 설계

1. 감성공학

(1) 감성공학의 정의
인간의 감성을 과학적으로 측정하고 평가한 것에 공학적 기술력을 결합시켜 새로운 제품을 만들어 인간에게 더욱 편리하고 안락할 수 있게 도모하려는 기술이다.

(2) 감성공학 기술 활용 분야

① 인간공학·인지공학 등 인간 특성을 파악하려는 연구에 기본을 둔 생체 측정 기술이다,
② 인간 특성에 적합하도록 사용자 인터페이스를 실현하기 위한 기술로서 센서 공학·퍼지 뉴럴 네트워크 기술·신경망 기술 등 인간의 오감(시각·청각·촉각·미각·후각) 센서 및 감성 처리 기술이다,
③ 사용성 평가 기술·가상현실 기술 등으로서 인간에 대한 적합성을 판단하고 새로운 감성을 창출하기 위한 기술이다.

(3) 나가마치 미츠오 교수의 감성공학 접근 방법

① 감성공학 1류 : 인간의 감성 이미지를 측정하는 방법이며, 이를 통해 제품에 대한 이미지를 조사·분석하여 제품의 디자인 요소와 연계시킨다.
② 감성공학 2류 : 개별적 특성과 생활 방식으로부터 개인이 갖고 있는 이미지를 구체화하는 방법이다. 감성의 심리적 특성을 강조한 접근 방법이라 할 수 있으며, 감성의 개인성에 중점을 둔 '문화적 감성'의 일부를 반영하기도 한다.
③ 감성공학 3류 : 공학적인 방법으로 접근하여 인간의 감각을 측정하고, 이를 바탕으로 수학적 모델을 구축하여 활용한다. 대상이 되는 제품의 물리적 특성과 인간의 감각이 객관화된 지표 사이의 연관성을 분석하여 제품 설계에 응용할 수 있으며, 측정 시 감성의 생리적 특성을 중시한다.

2. UI 설계 도구

- 와이어프레임(Wireframe) : UI 중심의 화면 레이아웃
- 목업(Mockup) : 실물과 흡사한 정적인 형태의 모형
- 프로토타입(Prototype) : 다양한 인터랙션이 결합되어 실제 서비스처럼 작동하는 모형
- 스토리보드(Storyboard) : 정책, 프로세스, 와이어프레임, 디스크립션 등이 모두 포함된 설계 문서

(1) 일반문서 도구
- 워드,엑셀,파워포인트...등

(2) 화면 설계 툴
① 카카오 오븐 (Oven)
② Power Mockup
③ 발사믹 Mockup

(3) 프로토타이핑 툴
① UX핀(UXPin)
② 액슈어(AXURE)
③ 네이버 프로토나우 (protoNow)

(4) ui 디자인 툴
① 스케치(Sketch)
② 어도비 익스피리언스 디자인 CC(Adobe XD)

(5) 디자인 산출물로 작업하는 프로토타이핑 툴
① 인비전(Invision)
② 픽사에이트(Pixate)
③ 프레이머(Framer)

