1과목 소프트웨어 설계

1-1 요구사항 확인

A 1-1-1 소프트웨어 생명주기

소프트웨어 공학의 기본 원칙

1. 최소비용으로 품질 높은 소프트웨어 상품 개발(현대 기술을 계속 적용)
2. 지속적으로 검증
3. 결과에 대한 명확한 기록 유지

Waterfall Model 폭포수 모형 (고전적 생명 주기 모형) – ~~검토>계획>분석>설계>구현>검사>유지보수~~

1. 한 단계가 끝나야만 다음 단계로 넘어갈 수 잇는 선형 순차적 모형
2. 적용 경험과 성공 사례가 많다
3. 결과물이 명확하게 산출되어야 함 / 단계적 정의와 산출물이 명확하다
4. ~~매뉴얼 작성~~
5. ~~두 개 이상의 과정이 병행하여 수행되지 않는다.~~

프로토타입 모형 (원형 모형) – ~~수집>설계>구축>평가>조정>구현~~

1. 사용자와 시스템 사이의 인터페이스에 중점을 두어 개발
2. 골격코드가 된다
3. 폭포수 모형의 단점을 보안하기 위한 모형

Spiral Model 나선형 모형 (점진적 모형) – 계획>분석>개발,검증>평가 -> 반복

1. 프로토타입을 지속적으로 발전시킴 / 대규모 시스템에 적합
2. 위험을 관리하고 최소화 하는 것을 목적으로 함
3. 누락되거나 추가된 요구사항을 첨가할 수 있다.
4. 요구사항이나 아키텍처를 이해하기 어렵다거나 중심이 되는 기술에 문제가 있는 경우 적합한 모델

애자일 모형 – ~~개발->설계->테스트 ->반복~~

1. 스크럼, XP, 기능 중심개발, 기능 주도 개발
2. 고객과의 소통에 초점을 맞춘 방법론
3. 고객의 평가와 요구를 적극 수용
4. 고객이 요구사항에 우선순위를 부여하여 개발 작업을 진행
5. 소규모 프로젝트, 숙달된 개발자, 급변하는 요구사항에 적합
6. 민첩한, 기민한 의미
7. 계획을 따리기보다는 변화에 대응
8. ~~기업 활동 전반에 걸쳐 사용~~

V 모델

1. Perry에 의해 제안
2. 세부적인 테스트 과정으로 구성되어 신뢰도 높은 시스템을 개발하는데 효과적
3. 개발 작업과 검증 작업 사이의 관계를 명확히 드러내 놓은 폭포수 모델의 변형
4. 폭포수 모델이 산출물 중심이라면 V 모델은 작업과 결과의 검증에 초점을 둔다.

B 1-1-2 스크럼 기법

스크럼 구성

1. ~~제품 책임자 (PO)~~
2. 스크럼 마스터 (SM)

   스크럼 프로세스를 따르고, 팀이 효과적으로 활용할 수 있도록 보장하는 역할

3. ~~개발팀 (DT)~~

개발 프로세스

1. 제품 백로그

   스크럼 팀이 해결해야 하는 목록으로 요구사항, 아키텍처 정의 등 포함

2. ~~스프린트 계획 회의~~
3. 스프린트

   2~4주 정도의 기간으로 결정해 작업을 진행

   속도(Velocity) : 한 팀이 어느 정도의 제품 백로그를 감당할 수 있는지에 대한 추정치로 볼 수 있다.

4. ~~일일 스크럼 회의~~
5. ~~스프린트 검토 회의~~
6. ~~스프린트 회고~~

A 1-1-3 XP(eXtreme Programming) 기법

XP 기법

1. 소규모 개발 조직이 불확실하고 변경이 많은 요구를 접하였을 때 적절한 방법
2. 상식적인 원리와 경험을 최대한 끌어 올리는 것
3. 구체적인 실천 방법을 정의하고 있으며, 개발 문서 보다는 소스코드에 중점을 둔다
4. 사용자의 요구사항은 언제든지 변할 수 있다.
5. 고객과 직접 대면하며 요구사항을 이야기하기 위해 사용자 스토리(User Story)를 활용할 수 있다.
6. 기존의 방법론에 비해 실용성(Pragmatism)을 강조한 것이라고 볼 수 있다.

핵심 가치

1. 의사소통
2. 단순성
3. Feedback 피드백 - 사용자가 명령에 대한 진행 상황과 표시된 내용 해석을 도움
4. ~~용기~~
5. ~~존중~~

기본 원리, 실천 방법

1. Pair Programming 짝 프로그래밍
2. Collective Ownership 공동 코드 소유
3. Test-Driven Development 테스트 주도 개발
4. Continuous Integration 계속적인 통합
5. ~~Whole Team 전체 팀~~
6. Refactoring 리팩토링, ~~Design Improvement 디자인 개선~~

   적은 비용으로 수정할 수 있도록 겉으로 보이는 독장의 변화 없이 내부 구조를 변경
7. ~~Small Releases 소규모 릴리즈~~

B 1-1-4 현행 시스템 파악 (분석)

현행 시스템 파악 절차 : DBMS, 네트워크, 운영체제

1. 단계

   시스템 구성 파악

   시스템 기능 파악

   시스템 인터페이스 파악

2. 단계

   아키텍처 구성 파악

   소프트웨어 구성 파악

3. 단계

   하드웨어 구성 파악

   네트워크 구성 파악

B 1-1-5 개발 기술 환경 파악

DBMS 관련 요구사항 식별 시 고려사항 5가지

   가용성, 성능, 상호 호환성, ~~기술지원, 구축비용~~

프로그래밍 언어 선택 시 고려할 사항

1. 개발 정보시스템의 특성
2. 사용자의 요구사항
3. 컴파일러의 가용성

WAS 종류

   JEUS, Tomcat, WebSphere, ~~GlassFish, JBoss, Jetty,Resin, WebLogic~~

A 1-1-6 요구사항 정의

기능 요구사항

1. 시스템이 무엇인지, 어떤 기능을 하는지에 대한 사항
2. 시스템의 입출력으로 무엇이 포함되어야 하는지
3. 시스템이 데이터를 저장하거나 연산을 수행해야 하는지
4. 시스템이 반드시 수행해야 하는 기능
5. 사용자가 시스템을 통해 제공받기를 원하는 기능

비기능 요구사항

1. 성능 요구사항 : 처리속도, 처리량, 동적 정적 적용량, 가용성 등
2. 데이터 요구사항 : 자료, 기술, 보안, 안전이 필요한 데이터를 구축하는데 필요한
3. 시스템 장비 구성 요구사항 : 하드웨어, 소프트웨어, 네트워크 등
4. 인터페이스 요구사항 : 시스템 사용자 인터페이스에 대한 정보교환에 사용되는 기술
5. 테스트 요구사항 : 테스트하고 점검하기 위한
6. 보안 요구사항 : 기능, 운영 접근을 통제하기 위한
7. 품질 요구사항 : 가용성, 정합성, 상호호환성, 대응성, 신뢰성, 사용성, 우지관리성, 이식성, 확장성, 보안성
8. 제약사항 : 기술, 표준, 업무 법 제도 등
9. 프로젝트 관리 요구사항 : 프로젝트의 원활한 수행을 위한 관리 방법에 대한
10. 프로젝트 지원 요구사항 : 프로젝트의 원활한 수행을 위한 지원 사항, 방안에 대한

사용자 요구사항

1. 사용자 관점에서 본 시스템이 제공해야 할 요구사항
2. 사용자를 위한 것으로 친숙한 표현으로 이해하기 쉽게 작성

시스템 요구사항

1. 개발자 관점에서 본 시스템 전체가 사용자와 다른 시스템에 제공해야 할 요구사항
2. 사용자 요구사항에 비해 전문적이고 기술적인 용어로 표현
3. 소프트웨어 요구사항이라고도 한다.

정형 명세 기법

1. 수학적인 원리, 모델 기반
2. ~~수학적 기호, 정형화된 표기법~~
3. 정확, 간결하게 표현
4. ~~일관성, 완전성 검증 가능~~
5. ~~사용자가 이해하기 어려움~~

비정형 명세 기법

1. ~~상태/기능/객체 중심~~
2. 명사, 동사 등의 자연어를 기반으로 서술, 다이어그램 작성
3. ~~일관성 떨어짐~~
4. ~~해석이 다를 수 있음~~
5. ~~내용 이해가 쉬어 의사소통 용이~~

요구사항 분석 – 요구사항 개발 프로세스의 순서 = 도출>분석>명세>확인

1. 요구 분석 - 사용자의 요구에 대해 이해하는 단계
2. 요구 추출 - 사용자의 요구를 찾는 단계
3. 도메인 분석 - 요구에 대한 정보를 수집하고 배경을 분석하여 모델링 함

요구사항 분석이 어려운 이유

1. 개발자와 사용자 간의 지식이나 표현의 차이가 크다.
2. 사용자의 요구사항이 모호하고 불명확하다.
3. 소프트웨어 개발 과정 중에 요구사항이 계속 변할 수 있다.

요구사항 검증

1. 고객이 정말 원하는 시스템을 제대로 정의하고 있는지 점검하는 과정
2. 개발 완료 후 문제점이 발견될 경우 재작업 비용이 들 수 있기 때문에 검증은 중요
3. 실제 요구를 반영하는지 문서상의 요구사항은 서로 상충되지 않는지 등을 점검
4. ~~내용이 이해하기 쉬운지, 일관성 있는지 검증~~
5. ~~이해관계자들이 검토~~

~~6. 요구사항 검증 과정을 통해 모든 문제를 확인할 수 있는 것은 아니다.~~

~~7. 요구사항 정의 문서들에 대해 형상관리를 수행~~

A 1-1-7 요구사항 분석

개요

1. 비용과 일정에 대한 제약 설정
2. 타당성 조사
3. 요구사항 정의 문서화

~~4. 사용자의 요구를 정확하게 추출하여 목표를 정함~~

~~5. 해결 방식을 결정~~

~~6. 소프트웨어 분석가에 의해 요구사항 분석이 수행되는 단계 = 요구사항 분석 단계~~

~~7. UML, 자료흐름도, 자료사전, 소단위 명세서, 개체 관계도, 상태 전이도, 제어 명세서~~

과정

1. 문서화를 통해 유지보수가 유용하게 활용
2. 자료흐름도, 자료사전 등 효과적으로 이용
3. 구체적인 명세를 위해 소단위 명세서가 활용

자료흐름도(DFD)

1. 자료 흐름 그래프, 버블(Bubble) 차트
2. 구조적 분석 기법
3. Process(원, 사각형), Terminator(사각형), Data Store[직선, 평행선(단선/이중선), Data Flow(화살표)
4. 프로세스(process), 단말(terminator), 자료흐름(data store), 자료저장소(data flow) 4가지

~~5. 단계적으로 세분화~~

~~6. 처리를 거쳐 변환될 때마다 새로운 이름이 부여~~

~~7. 처리는 입력 자료가 발생하면 기능을 수행한 후 출력 자료를 산출~~

자료 사전(DD) 기호

1. 자료의 정의 =
2. 자료의 연결 + 그리고and
3. 자료의 생략 ( )
4. 자료의 선택 [|] 또는or
5. 자료의 반복 { }
6. 자료의 설명 ** 주석

B 1-1-8 요구사항 분석 CASE와 HIPO

종류

1. SADT : SoftTech사에서 개발, 블록 다이어그램을 채택한 자동화 도구
2. SREM
3. PSL/PSA
4. TAGS

HIPO

1. 가시적 도표, 총체적 도표, 세부적 도표
2. 기능과 자료의 의존 관계를 동시에 표현
3. 보기 쉽고 이해하기 쉽다.
4. 하향식 소프트웨어 개발을 위한 문서화 도구
5. ~~입력, 처리, 출력으로 구성~~
6. ~~체계적 문서 관리 가능~~
7. ~~변경, 유지보수가 용이~~
8. ~~시스템의 기능을 여러 개 고유 모듈들로 분할하여 인터페이스를 계층 구조로 표현~~

A 1-1-9 UML

언어

1. 객체 지향 시스템을 개발할 때 산출물을 명세화, 시각화, 문서화 하는데 사용
2. 개발하는 시스템을 이해하기 쉬운 형태로 표현하는 분석가, 의뢰인, 설계자가 효율적인 의사소통을 제공
3. 개발 방법론, 개발 프로세스가 아니라 표준화된 모델링 언어이다.

기본 구성 요소

1. 사물 : Things
2. 관계 : Relationships
3. 다이어그램 : Diagram

관계 (화살표)

1. 실선 - 삼각 : 일반화
2. 점선 - 꺽쇠 : 의존 Dependency
    A. 명세가 바뀌면 다른 사물에 영향을 줌
    B. 한 클래스가 다른 클래스를 오퍼레이션의 매개변수로 사용
3. 점선 – 삼각 : Realization 실체화
    A. 한 객체가 다른 객체에게 오퍼레이션을 수행하도록 지정하는 의미적 관계

다이어그램

1. 구조적 다이어그램 종류 (Structural Diagram) - 정적 모델
    A. 클래스 다이어그램 Class
    B. 컴포넌트 다이어그램 Component - 재사용되는 모든 단위, 인터페이스를 통해서만 접근
    C. 패키지 다이어그램 Package
    D. 배치 다이어그램 Deployment
    ~~E. 복합체 구조 다이어그램 Composite~~
    ~~F. 객체 다이어그램 Object~~
2. 행위 다이어그램 종류 - 동적 모델

A. 유스케이스 다이어그램 (기능적 모델)

B. 순차(시퀀스) 다이어그램

C. 상태 다이어그램 State

i. 럼바우(Rumbaugh) 객체지향 분석 기법에서 동적 모델링에 활용

ii. 상호 작용에 따라 상태가 변화하는지 표현

D. 활동(액티비티) 다이어그램

~~E. 커뮤니케이션 다이어그램~~

~~F. 상호작용 개요 다이어그램~~

~~G. 타이밍 다이어그램~~

스테오 타입

1. <<include>> : 포함 관계
2. <<extend>> : 확장 관계
3. <<interface>> : 인터페이스 정의
4. <<exception>> : 예외 정의
5. <<constructor>> : 생성자 역할

A 1-1-10 주요 UML 다이어그램

유스케이스 다이어그램 구성요소

1. 시스템 범위
2. 액터
    A. 시스템과 상호 작용하는 사람이나 다른 시스템에 의한 역할이다
    B. 주액터 : 시스템을 사용함으로써 이득을 얻는 대상을 의미
    C. 부액터 : 본 시스템과 데이터를 주고받는 연동 시스템
3. 유스케이스
    A. 사용자의 요구를 분석하는 데 사용 / 사용자 측면에서의 요구사항
    B. 사용자가 원하는 목표를 달성하기 위해 수행할 내용을 기술
4. 관계
    A. 연관, 확장(특별조건), 일반화, ~~포함~~

클래스 다이어그램 구성요소

1. 클래스
    A. Operation 오퍼레이션 :
        i. 클래스의 동작을 의미
        ii. UML에서 동작에 대한 인터페이스를 지칭한다고 볼 수 있다.
        iii. 클래스가 수행할 수 있는 동작, 함수, 메소드
    ~~B. 속성 : 클래스의 상태나 정보를 표현~~
    ~~C. 3개의 구획으로 나눠 클래스의 이름, 속성, 오퍼레이션을 표기~~
    ~~D. 클래스는 각각의 객체들이 갖는 속성과 오퍼레이션을 표현~~
2. 제약조건
3. 관계
    A. 클래스와 클래스 사이의 연관성 표현
    B. 클래스의 속성 사이의 관계

C. 클래스의 정적 구조를 표현

~~D. 연관, 집합, 포함, 일반화, 의존~~

순차(시퀀스) 다이어그램 구성요소

1. 생명선
2. 실행
3. 메시지
4. ~~액터~~
5. ~~객체~~

순차(시퀀스) 다이어그램 설명

1. 객체들의 상호 작용을 나타내기 위해 사용
2. 시간의 흐름에 따라 객체들이 주고 받는 메시지의 전달 과정을 강조
3. 교류 다이어그램의 한 종류로 볼 수 있다.
4. 회귀 메시지, 제어블록 등으로 구성
5. 다이어그램의 수직 방향이 시간의 흐름

1-2 화면 설계

A 1-2-1 사용자 인터페이스 User Interface

기본원칙

1. 직관성 : 누구나 쉽게 이해하고 사용할 수 있어야 한다.
2. ~~학습성~~ : 누구나 쉽게 배우고 익힐 수 있어야 한다.
3. ~~유연성~~ : 사용자의 요구사항을 최대한 수용하고 실수를 최소화해야 한다.
4. ~~유효성 : 사용자의 목적을 정확하고 완벽하게 달성해야 한다.~~

설계 지침

1. 사용자 중심 : 이해하기 편하고 쉽게 사용할 수 있는 환경을 제공
2. 가시성 : 주요 기능을 메인 화면에 노출하여 조작이 쉽도록 하여야 한다.
3. 접근성 : 사용자의 직무, 연령, 성별 등 다양한 계층을 수용하여야 한다.
4. 오류 발생 해결 : 오류가 발생하면 사용자가 쉽게 인지, 쉽게 수정할 수 있도록 설계
5. ~~사용성~~ : 효율성을 높이게 설계해야 한다..
6. ~~일관성 : 조작 방법 등을 일관성 있게 제공~~
7. ~~단순성 : 조작 방법을 단순화시켜 인지적 부담을 감소~~
8. ~~결과 예측 가능 : 작동시킬 기능만 보고도 결과를 미리 예측할 수 있게 설계~~
9. ~~심미성 : 디자인적으로 완성도 높게 글꼴이나 색상을 적용~~
10. ~~표준화 : 기능 구조와 디자인을 표준화~~
11. ~~명확성 : 개념적으로 쉽게 인지할 수 있도록 설계~~

사용자 인터페이스 구분

1. CLI : DOS, Unix 등의 운영체제에서 조작하기 위해 사용함
2. NUI : 멀티 터치, 동작 인식 등 사용자의 자연스러운 움직임을 인식하여 서로 주고받는 정보를 제공
3. ~~GUI~~
4. ~~VUI~~

~~5. OUI~~

사용자 인터페이스 개발 시스템의 기능

1. 사용자 입력의 검증
2. 에러 처리와 에러 메시지 처리
3. 도움과 프롬프트 제공

주요 모바일 제스처

1. Pinch : 두 손가락으로 넓히기/좁히기
2. Press : 오래 누르기
3. Flick : 빠르게 스크롤

~~4. Tap : 누르기~~

~~5. Double Tap : 두번 누르기~~

~~6. Drag누른 채 움직임~~

~~7. Pan : 누른 채 계속 움직임~~

B 1-2-3 UI 설계 도구

목업 ~~Mockup~~

1. 디자인, 사용방법 설명, 평가 등을 위해 실제화면과 유사하게 만든 정적인 형태의 모형
2. 시작적으로만 구성 요소를 배치하는 것으로 일반적으로 실제로 구현되지 않음

~~와이어프레임, 스토리보드, 프로토타입, 유스케이스~~

A 1-2-5 품질 요구사항

ISO/IEC 25000 = 소프트웨어 품질 관련 국제 표준

1. 소프트웨어 품질 평가를 위한 소프트웨어 품질평가 통합모델 표준
2. SQuaRE라고도 한다.
3. ~~ISO/IEC 12119~~ + ISO/IEC 9126 + ISO/IEC 14598을 통합

ISO/IEC 25010 = ISO/IEC 9126 + 보안성 + 호환성 = ~~소프트웨어 품질의 정의~~

테스트 국제 표준 (ISO/IEC 12119)

소프트웨어 품질 측정의 특성 (ISO/IEC 25010)

1. 기능성
    A. 정확성, ~~완전성, 적절성~~
2. 보안성
    A. 무결성, ~~기밀성, 부인방지, 책임추적성, 인증성~~
3. 사용성 = 쉽게 배우고 사용할 수 있는 정도
    A. ~~인지정도. 학습성. 조작성. 사용자 오류 방지. UI 미학. 접근성~~
4. 신뢰성 = 주어진 시간동안 주어진 기능을 오류 없이 수행하는 정도
    ~~A. 성숙성, 사용가능성, 결함 허용성, 복구성~~
5. 이식성 = 하나 이상의 하드웨어 환경에서 운용되기 위해 쉽게 적용할 수 있는지 정도
    ~~A. 적응성, 설치성, 대체성~~
6. ~~호환성~~
    ~~A. 공존성, 상호 운영성~~
7. ~~효율성~~
    ~~A. 시간 효율성, 자원 효율성, 사양~~
8. ~~유지보수성~~
    ~~A. 모듈성, 재사용성, 분석성, 변경성, 시험성~~

B 1-2-9 UI 상세 설계

UI 요소

1. 라디오 버튼 : 여러 개의 선택 항목 중 하나의 선택만 가능

2. ~~체크 박스~~

3. ~~텍스트 박스~~

4. ~~콤보 상자~~

5. ~~목록상자~~

1-3 애플리케이션 설계

A 1-3-1 소프트웨어 아키텍처

소프트웨어 상위 설계

1. 별칭 : 아키텍처 설계, 예비 설계, 인터페이스 정의, 사용자 인터페이스 설계
2. ~~설계 대상 : 시스템의 전체적인 구조~~
3. 세부 목록 : 구조, DB, 인터페이스

소프트웨어 하위 설계

1. ~~별칭 : 모듈 설계, 상세 설계, 인터페이스 작성~~
2. ~~설계 대상 : 시스템의 내부 구조 및 행위~~
3. ~~세부 목록 : 컴포넌트, 자료 구조, 알고리즘~~

모듈화

1. 프로그래밍 언어에서 Subroutine, Function 등으로 표현될 수 있다.
2. 시스템을 지능적으로 관리
3. 복잡도 문제를 해결하는데 도움
4. 유지보수와 수정을 용이하게 함.
5. 오류의 파급 효과를 최소화
6. 프로그램의 효율적인 관리

추상화

제어, 과정, 자료(데이터) = 제과자

정보은닉

1. 필요하지 않은 정보는 접근할 수 없다.
2. 모듈들 사이의 독립성 유지
3. IP주소와 같은 물리적 코드, 상세 데이터 구조 등 있다.

소프트웨어 아키텍처 품질속성

시스템 : 가용성, 변경 용이성, 사용성, ~~성능, 보안, 기능성, 확장성, 기타 속성~~

~~비즈니스 : 시작 적시성, 비용과 혜택, 예상 시스템 수명, 기타 속성~~

~~아키텍처 : 개념적 무결성, 정확성, 완결성, 구축 가능성, 기타 속성~~

소프트웨어 아키텍처의 설계 과정

1. 설계 목표 설정
2. 시스템 타입 결정
3. 스타일 적용 및 커스터마이즈 (아키텍처 패턴 적용)
4. 서브시스템의 기능, 인터페이스 동작 작성 (서브시스템 구체화)
5. 아키텍처 설계 검토 (검토)

협약에 의한 설계

컴포넌트 설계 시 "협약에 의한 설계"을 따를 경우, 해당 명세서에는

1. 컴포넌트의 오퍼레이션 사용 전에 참이 되어야 할 선행조건
2. 사용 후 만족 되어야 할 결과조건
3. 오퍼레이션이 실행되는 동안 항상 만족되어야 할 불변조건

A 1-3-2 아키텍처 패턴

파이프 필터 패턴

서브시스템이 입력 데이터를 받아 처리 후 결과를 다음 서브시스템으로 넘기는 과정 반복

MVC 패턴

1. 사용자 인터페이스를 담당하는 계층의 응집도를 높일 수 있다.
2. 여러 개의 다른 UI를 만들어 그 사이에 결합도를 낮출 수 있다.
3. 뷰 : 모델에 있는 데이터를 사용자 인터페이스에 보이는 역할을 담당.
4. 제어 : 모델에 명령을 보냄으로써 모델의 상태를 변경

Model 모델

1. 개발 대상을 추상화하고 기호나 그림 등으로 시각적으로 표현
2. 소프트웨어에 대한 이해도를 향상시킬 수 있다.
3. 이해 당사자 간의 의사소통이 향상된다.
4. 향후 개발될 시스템을 유추하기 위해서 하는 활동
5. ~~시스템 개발자가 실행~~

마스터 – 슬레이브 패턴

1. 실시간 시스템에서 사용
2. 마스터 프로세스는 연산, 통신, 조정을 책임진다.
3. 마스터 프로세스는 슬레이브 프로세스들을 제어 가능

A 1-3-3 객체 지향

객체

1. 상태, 동작, 고유 식별자를 가진 모든 것
2. 필요한 자료 구조와 수행하는 함수들을 가진 하나의 독립된 존재
3. 객체의 상태는 속성값에 의해 정의
4. 실세계에 존재하거나 생각할 수 있는 것

Class 클래스

1. 하나 이상의 유사한 객체들을 묶어 공통된 특성을 표현한 데이터 추상화
2. 각각의 객체를 인스턴스(Instance)라 한다

Encapsulation (캡슐화)

1. 연관된 데이터와 함수를 함께 묶어 외부와 경계를 만들고 필요한 인터페이스만을 밖으로 드러내는 과정
2. 속성과 관련된 연산(Operation)을 클래스 안에 묶어서 하나로 취급하는 것을 의미하는 객체지향 개념

Inheritance (상속)

1. 상위 클래스의 메소드와 속성을 하위 클래스가 물려받는 것

다형성

1. 상속받은 여러 개의 하위 객체들이 다른 형태의 특성을 갖는 객체로 이용 될 수 있는 성질
2. 현재 코드를 변경하지 않고 새로운 클래스를 쉽게 추가
3. 여러 가지 형태를 가지고 있다는 의미
4. 여러 가지 형태를 받아들일 수 있는 특징
5. 오버라이딩은 상위 클래스에서 정의한 일반 메소드의 구현을 하위 클래스에서 무시하고 재정의

연관성

1. 집단화 : 부분-전체, 부분-부분 관계
2. ~~연관화, 분류화, 일반화, 특수화, 상세화~~

A 1-3-4 객체 지향 분석 및 설계

객체지향 분석

소프트웨어를 개발하기 위한 비즈니스(업무)를 객체와 속성, 클래스와 멤버, 전체와 부분 등으로 나누어서 분석해 내는 기법

객체지향 분석의 방법론

1. Coad – Yourdon : E-R 다이어그램을 사용하여 객체의 행위를 데이터 모델링하는데 초점을 둔 방법
2. Rumbaugh : 객체, 정보(객체, Object) > 동적(상태, Dynamic) > 기능(DFD, 자료흐름도, Function) 모델링
3. ~~Booch : 미시적, 거시적~~
4. ~~Jacobson : 유스케이스~~
5. ~~Wirfs – Brock~~

설계 원칙 (클래스 설계원칙)

1. ISP 인터페이스 분리 원칙
   A. 클라이언트는 자신이 사용하지 않는 메소드와 의존관계를 맺으면 안 된다.
   B. 클라이언트는 사용하지 않는 인터페이스 때문에 영향을 받아서는 안 된다
2. OCP 개방-폐쇄 원칙 : 클래스는 확장에 대해 열려 있어야 하며 변경에 대해 닫혀 있어야 한다.
3. LSP 리스코프 치환 원칙 : 자식 클래스는 최소한 부모 클래스에서 가능한 행위는 수행할 수 있어야 함
4. SRP 단일 책임 원칙 : 하나의 클래스만 변경 가능 해야 한다.
5. DIP 의존 역전 원칙 : 클라이언트는 자신이 사용하는 메소드와 의존관계를 갖지 않도록 해야 한다

A 1-3-5 모듈

모듈의 개요

1. 다른 것들과 구별될 수 있는 독립적인 기능을 가진 단위(Unit)
2. 독립적인 컴파일 가능
3. 유일한 이름
4. 상호 접근이 가능

모듈화

1. 각각의 모듈을 별개로 만들고 수정할 수 있기 때문에 좋은 구조가 된다
2. 모듈 간의 결합도가 약해야 독립적인 모듈이 된다.
3. 구성 요소들 간의 응집도가 강해야 좋은 모듈 설계

2장 문제) 모듈 특징

1. 소프트웨어 구조를 이룬다
2. 다른 것들과 구별될 수 있는 독립적인 기능을 갖는 단위.
3. 명령어들의 집합
4. 서로 모여 하나의 완전한 프로그램으로 만들어질 수 있다.

결합도 – 내용(Content)>공통(Common)>외부>제어(Control)>스태프(Stamp)>자료(Data) – 내공외자스자

1. 결합도 : 두 모듈 간의 상호작용, 의존도 정도를 나타냄
2. 내용 : 하나의 모듈이 직접적으로 다른 모듈의 내용을 참조할 때 두 모듈은 내용적으로 결합
3. 공통 : 두 모듈이 동일한 전역 데이터를 접근
4. ~~외부 :~~
5. 제어 :
    A. 어떤 모듈이 다른 모듈의 내부 논리 조직을 제어하기 위한 목적으로 제어 신호를 통하여 통신
    B. 하위 모듈에서 상위 모듈로 제어 신호가 이동하여 상위 모듈에게 처리 명령을 부여
6. 스태프 : 자료 구조 형태로 전달됨
7. ~~자료 :~~

응집도 – 기능적(Functional)>순차적(Sequential)>교환적>절차적>시간적>논리적(Logical)>우연적(Coincidental)

1. ~~기능적 :~~
2. ~~순차적 : Sequential~~
3. ~~교환적 :~~
4. 절차적 : 모듈 안의 구성 요소들이 그 기능을 순차적으로 수행할 경우
5. Temporal 시간적 : 모듈 내 구성 요소들이 서로 다른 기능을 같은 시간대에 함께 실행
6. ~~논리적 : Logical~~
7. Coincidental 우연적 : 서로 간에 어떠한 의미 있는 연관 관계도 지니지 않는 기능 요소로 구성

팬인 / 팬아웃

1. 팬인 = 상위 개수 / 모듈이 들어오는 개수
2. 팬아웃 = 하위 개수 / 모듈이 나가는 개수

N-S 차트

1. 논리의 기술에 중점을 둔 도형식 표현 방법
2. 연속, 선택, 다중선택, 반복 등의 제어 논리 구조로 표현
3. 조건이 복합되어 있는 곳의 처리를 시각적으로 명확히 식별하는데 적합
4. 이해하기 쉽고 코드 변환이 용이

A 1-3-6 공통 모듈

공통 모듈 개요

1. 명확성 : 해당 기능에 대해 일관되게 이해되고 한 가지로 해석될 수 있도록 작성하는 원칙
2. 정확성
3. 완전성
4. 일관성
5. 추적성

재사용 : 소프트웨어의 일부분을 다른 시스템에서 사용할 수 있는 정도

1. 함수와 객체
2. 컴포넌트 : 인터페이스를 통해서만 접근
3. 애플리케이션

효과적인 모듈 설계 방안 (S/W 설계 지침)

1. 적당한 모듈의 크기를 유지
2. 이식성 고려
3. 모듈의 기능을 예측할 수 있도록 정의
4. 모듈 간의 접속 관계를 분석하여 복잡도와 중복을 줄인다
5. 모듈 간의 효과적인 제어를 위해 설계에서 계층적 자료 조직이 제시되어야 한다.
6. 결합도는 줄이고 응집도는 높여서 모듈의 독립성과 재사용성 높인다.
7. 복잡도와 중복성을 줄이고 일관성을 유지
8. 유지보수가 용이
9. 모듈의 기능은 예측이 가능해야 하며 지나치게 제한적이어서는 안된다.
10. ~~모듈의 제어 영역 안에서 그 모듈의 영향 영역을 유지~~
11. ~~모듈 크기는 시스템의 전반적인 기능과 구조를 이해하기 쉬운 크기로 분해~~
12. ~~하나의 입구와 하나의 출구를 갖도록 함~~
13. ~~모듈 인터페이스를 설계 , 계층적 관계를 정의하는 자료가 제시~~

A 1-3-7 코드

코드 개요

1. 식별 기능
2. 분류 기능
3. 표준화 기능
4. ~~배열 기능~~
5. ~~간소화 기능~~

코드 종류

1. 순차 코드 : 일정한 일련번호를 부여
2. 표의 숫자 코드 : 코드화 대상 항목의 중량, 면적, 용량 등의 물리적 수치를 이용
3. ~~블록 코드~~
4. ~~10진 코드~~
5. ~~그룹 분류 코드~~
6. ~~연상 코드~~
7. ~~합성 코드~~

A 1-3-8 디자인 패턴 (GoF)

디자인 패턴 개요

소프트웨어 설계에서 자주 발생하는 문제에 대한 일반적이고 반복적인 해결방안

장 단점

1. 소프트웨어 구조 파악이 용이, 코드의 품질 향상
2. 객체지향 설계 및 구현의 생산성을 높이는데 적합
3. 재사용을 위한 개발 시간 단축
4. 개발자간 원활한 의사소통 가능
5. ~~초기 투자 비용 부담~~
6. ~~설계변경 요청에 대한 유연한 대처 가능~~
7. ~~다른 기반의 애플리케이션 개발에 적합하지 않다.~~

생성 패턴

1. Factory 팩토리 메소드 :
   A. 상위클래스에서 객체를 생성하기 위한 인터페이스를 정의
   B. 하위클래스에서 인스턴스 생성 / 어떤 클래스가 인스턴스화 될 것인지는 서브 클래스가 결정
2. prototype 프로토타입 : 프로토타입을 먼저 생성하고 인스턴스를 복제하여 사용
3. Singleton 싱글턴 : 특정 클래스의 인스턴스가 오직 하나임을 보장, 인스턴스에 대한 접근 방법을 제공
4. 추상 팩토리 Abstract
5. 빌더 Builder

구조 패턴

1. Adaoter 어댑터
2. Bridge 브리지
3. proxy 프록시
4. ~~컴포지트~~
5. ~~데코레이터~~

~~6. 퍼싸드~~

~~7. 플라이웨이트~~

행위 패턴 : 클래스, 객체들이 상호작용하는 방법과 책임을 분산하는 방법을 정의

mediator 중재자 : 객체간의 통제와 지시의 역할을 하는 중재자를 두어 객체지향의 목표를 달성하게 함

커맨드, 옵서버, 상태, ~~전략, 방문자, 책임 연쇄, 인터프리터, 반복자, 메멘토, 템플릿 메소드~~

1-4 인터페이스 설계

A 1-4-2 인터페이스 요구사항 검증

요구사항 검증 방법

1. 동료검토
    A. 요구사항 명세서 작성자가 요구사항 명세서를 설명, 이해관계자들이 설명을 들으면서 결함을 발견
2. 워크스루
    A. 검토 회의 전에 미리 배포하여 사전 검토 후 짧은 회의를 통해 오류를 조기에 검출하는 목적
    B. 사용사례를 확장하여 명세하거나 설계 다이어그램, 원시코드, 테스트 케이스 등에 적용
    C. 복잡한 알고리즘, 반복, 실시간 동작, 병행 처리의 기능, 동작을 이해하려고 할 때 유용
    D. 단순한 테스트 케이스를 이용하여 프로덕트를 수작업으로 수행
3. 인스펙션 - 계획>사전교육>준비>인스펙션 회의>수정>후속조치
    A. 프로그램을 수행시켜보는 것 대신에 읽어보고 눈으로 확인하는 방법으로 볼 수 있다.
    B. 코드 품질 향상 기법 중 하나이다.
    C. 결함과 함께 코딩 표준 준수 여부, 효율성 등의 다른 품질 이슈를 검사하기도 한다.

B 1-4-5 인터페이스 방법 명세화

시스템 연계 기술

1. 소켓(Socket) 기술 :

   통신을 위한 프로그램을 생성하여 포트를 할당하고,

   클라이언트의 통신 요청 시 클라이언트와 연결하는 내외부 송수신 연계 기술

오류 식별 및 처리 방안 명세화

1. 송신 시스템

   시스템 인터페이스를 구성하는 시스템으로,

   연계할 데이터를 데이터베이스와 애플리케이션으로부터

   연계 테이블 또는 파일 형태로 생성하며 송신하는 시스템

2. ~~연계 서버~~

3. ~~수신 시스템~~

A 1-4-7 미들웨어 솔루션 명세

미들웨어 개념

1. ~~미들 + 소프트웨어 합성어~~
2. 클라이언트-서버 간의 통신을 담당하는 시스템 소프트웨어
3. 분산 컴퓨팅 환경에서 서로 다른 기종간 통신환경을 연결하여 원만한 통신이 이루어지도록 서비스
4. 분산 시스템에서 다양한 부분을 관리하고 통신하며 데이터를 교환
5. 위치 투명성 제공
6. 분산 시스템의 여러 컴포넌트가 요구하는 재사용 가능한 서비스의 구현을 제공
7. 여러 운영체제에서 응용 프로그램들 사이에 위치한 소프트웨어
8. 소프트웨어 컴포넌트를 연결하기 위한 준비된 인프라 구조 제공
9. 1:1, 1:N, N:N 등 연결
10. 종류 : DB, RPC, MOM, TP-Monitor, ORB, WAS
11. 미들웨어 내부 동작을 확인하려면 별도의 응용 소프트웨어를 사용
12. ~~표준화된 인터페이스를 제공함으로써 시스템 간의 데이터 교환에 일관성 보장~~

RPC

1. 원격 프로시저를 호출하는 방식

MOM 메시지 지향 미들웨어

1. 독립적인 애플리케이션을 하나의 통합된 시스템으로 묶기 위한 역할
2. 송신측과 수신측의 연결 시 메시지 큐를 활용하는 방법
3. 상이한 애플리케이션 간 통신을 비동기 방식으로 지원

TP-Monitor

1. 트랜잭션(Transaction)을 처리(Processing)하고 데이터를 감시(Monitoring)하고 제어

2과목 소프트웨어 개발

2-1 데이터 입출력 구현

A 2-1-1 자료 구조

자료 구조의 분류

1. 선형
    A. 배열
    B. 선형 리스트
    C. 스택
    D. 큐
    E. 데크
2. 비선형
    A. 트리 : 노드와 선분으로 구성, 정점 사이에 사이클(Cycle)이 없음
    B. 그래프

선형리스트

1. 연속(Contiguous) 리스트
2. 연결(Linked) 리스트
    A. 노드의 삽입이나 삭제가 쉽다
    B. 연결을 해주는 포인터를 위한 추가 공간이 필요하다
    C. 연결 리스트 중에서 중간 노드 연결이 끊어지면 그 다음 노드를 찾기 힘들다

Stack 스택

1. 후입선출(LIFO) 방법을 사용
2. 입출력이 한쪽 끝으로만 제한된 리스트

3. Underflow : Top의 주소가 0일 때 / 더 이상 삭제할 데이터가 없는 상태에서 데이터를 삭제하면 발생

~~4. Overflow : Top의 주소가 M보다 클 때~~

스택을 이용한 연산

1. 재귀 호출
2. 후위 표현(Post-fix)의 연산 (후위 표기법)
3. 깊이우선탐색(DFS) : 중앙 > 좌측으로 깊이 내려감 > 나머지

스택의 응용 분야

1. 서브루틴 호출
2. 인터럽트 처리
3. 수식 계산 및 수식 표기법을 응용

큐(Queue)

1. 선입선출(FIFO)

방향/무방향 그래프의 최대 간선 수

1. 무방향 그래프의 최대 간선 수 : n(n-1)/2

~~2. 방향 그래프의 최대 간선 수 : n(n-1)~~

A 2-1-2 트리(Tree)

트리 개요

1. 차수(디그리 Degree) : 하위 노드의 개수
2. 단말 노드 : 하위 노드가 없는 노드 (차수 0개인 노드)

트리 운행법

1. Preorder(전위) : 중 > 좌 > 우
2. Inorder(중위) : 좌 > 중 > 우
3. Postorder(후위): 좌 > 우 > 중

수식의 표기법

1. Preorder(전위) : 연산자 > 좌 > 우
2. Inorder(중위) : 좌 > 연산자 > 우
3. Postorder(후위) : 좌 > 우 > 연산자

A 2-1-3 정렬(Sort)

삽입 정렬

1. 1번과 2번을 확인
2. 2번과 3번을 확인
3. 4번과 1,2,3번을 확인
4. 5번과 전부 확인

선택 정렬 Selection

1. 1번자리에 맞는 거 앞에부터 찾아 바꾸기 반복
2. 2번자리에 맞는 거 앞에부터 찾아 바꾸기 반복
3. 3번자리에 맞는 거 앞에부터 전부 찾아 바꾸기 반복
4. 4번자리에 맞는 거 앞에부터 전부 찾아 바꾸기 반복

버블 정렬

1. 1pass 크기비교 : 1번과 2번 확인 > 2번과 3번 확인 > 3번과 4번 확인 > 4번과 5번 확인
2. 2pss 맨 뒤에서 한자리 고정 : 1번과 2번 확인 > 2번과 3번 확인 > 3번과 4번 확인
3. 3pss 맨 뒤에서 한자리 고정 : 1번과 2번 확인 > 2번과 3번 확인
4. 1번과 2번 확인

Quick 퀵 정렬

1. 레코드의 많은 자료 이동을 없애고 하나의 파일을 부분적으로 나누어 가면서 정렬
2. 분할 정복에 기반한 알고리즘으로 피벗을 사용

힙 정렬

1. 정렬할 입력 레코드들로 힙을 구성하고 가장 큰 키 값을 갖는 루트 노드를 제거하는 과정 반복
2. 평균 수행 시간 'O'이다
3. 완전 이진 트리로 입력 자료의 레코드를 구성

2-Way 합병 정렬

1. 정렬된 Nrodml 데이터를 처리하는데 O의 시간이 소요되는 정렬 알고리즘

B 2-1-4 검색 – 이분 검색/해싱

이분(이진) 검색

1. 탐색 효율이 좋고 탐색 시간이 적게 소요
2. 비교 횟수를 거듭할 때마다 검색 대상이 되는 데이터의 수가 절반으로 줄어듦
3. 검색할 데이터가 정렬되어 있어야 함
4. 방법 :
    A. 중앙값을 찾으려는 값과 비교 (좌우를 찾음)
    B. 중앙값을 찾으려는 값과 비교 (좌우를 찾음)
    C. 중앙값을 찾으려는 값과 비교 (좌우를 찾음)
    D. 반복

해싱

1. 해시 테이블
2. 해시 함수
    A. 제산법
    B. 제곱법
    C. 숫자 분석법
    D. 폴딩법 : 해싱 함수 중 레코드 키를 여러 부분 나누고, 나눈 부분을 더하거나 XOR 값을 사용
    ~~E. 기수 변환법~~
    ~~F. 대수적 코딩법~~
    ~~G. 무작위법~~

B 2-1-5 데이터베이스 개요

스키마

1. 내부 스키마 : 물리적 저장장치와 관계가 깊은 스키마
2. 외부 스키마 : 데이터베이스의 논리적 구조를 정의
3. 개념 스키마 : 데이터베이스 전체 정의 / 데이터 개체, 관계, 제약 조건 등을 명세

B 2-1-7 절차형 SQL

테스트와 디버그의 목적

1. 테스트 : 오류를 찾는 작업
2. 디버그 : 오류를 수정하는 작업

2-2 통합 구현

B 2-2-2 단위 모듈 테스트

테스트 케이스

1. 식별자 : 항목 식별자, 일련번호
2. 테스트 항목 : 테스트 대상(모듈, 기능)
3. 입력 명세 : 테스트 조건, 입력(테스트) 데이터
4. 출력 명세 : 예상 결과
5. 환경 설정 : 하드웨어, 소프트웨어
6. 특수 절차 요구 : 특별 요구 절차
7. 의존성 기술 : 의존성

B 2-2-3 개발 지원 도구

IDE(통합개발환경) 도구의 기능

1. Coding(코딩) : 프로그래밍 언어를 가지고 컴퓨터 프로그램을 작성할 수 있는 환경을 제공
2. Debugging(디버깅) : 프로그램에서 발견되는 버그를 찾아 수정할 수 있는 기능
3. Deployment(배포) : 소프트웨어를 최정 사용자에게 전달하기 위한 기능
4. Compile(컴파일) : 개발자 언어를 컴퓨터 언어로 변환

2-3 제품 소프트웨어 패키징

A 2-3-1 소프트웨어 패키징

소프트웨어 패키징 개요

1. 모듈화하여 패키징
2. 고객의 편의성을 위해 매뉴얼 및 버전관리를 지속적으로 한다
3. 범용 환경에서 사용이 가능하도록 일반적인 배포 형태로 패키징
4. 개발자가 아닌 사용자 중심으로 진행

패키징 시 고려사항

1. 반드시 내부 콘텐츠에 대한 암호화 및 보안을 고려
2. 사용자 편의성을 위한 복잡성 및 비효율성 문제를 고려
3. 제품 소프트웨어 종류에 적합한 암호화 알고리즘을 적용
4. 다른 콘텐츠 연동을 고려
5. ~~Managed Service 형태로 제공~~
6. ~~UI는 시각적인 자료와 함께 제공하고 매뉴얼과 일치시켜 패키징~~
7. ~~최소환경을 정의~~

A 2-3-3 디지털 저작권 관리(DRM)

디지털 저작권 관리

1. 접근 제어 기술
2. 사용권한 관리, 과금, 유통 단계를 관리하는 기술
3. 클리어링 하우스는 콘텐츠 라이선스를 발급하고 권한을 부여하는 시스템

디지털 저작권 관리의 흐름 및 구성 요소

1. 콘텐츠 제공자 : 콘텐츠를 제공하는 저작권자
2. 클리어링 하우스 : 키 관리 및 라이선스 발급 관리
3. DRM Controller (DRM 컨트롤러) : 배포된 콘텐츠의 이용 권한 통제
4. Contents Distributor (콘텐츠 분배자) : 암호화된 콘텐츠를 유통하는 곳이나 사람
5. Packager (패키저) : 콘텐츠를 메타 데이터와 함께 배포 가능한 단위로 묶는다.
6. 보안 컨테이너 : 콘텐츠 원본을 안전하게 유통하기 위한 전자적 보안 장치

디지털 저작권 관리의 기술 요소

1. 암호화 : 콘텐츠 및 라이선스를 암호화하고 전자 서명을 할 수 있는 기술
2. 키 관리 : 콘텐츠를 암호화한 키에 대한 저장 및 분배 기술
3. 식별 기술 : 콘텐츠에 대한 식별 체계 표현 기술
4. 정책 관리 : 라이선스 발급 및 관리 기술
5. 크랙 방지 : 크랙에 의한 콘텐츠 사용 방지 기술
6. ~~암호화 파일 생성 : 콘텐츠를 암호화된 콘텐츠로 생성하기 위한 기술~~
7. ~~저작권 표현 : 라이선스의 내용 표현 기술~~
8. ~~인증 : 라이선스 발급 및 사용의 기준이 되는 사용자 인증 기술~~

B 2-3-4 소프트웨어 설치 매뉴얼 작성

소프트웨어 설치 매뉴얼의 개요

1. 설치 과정에서 표시될 수 있는 예외 상황에 관련 내용을 별도로 구분하여 설명
2. 설치 시작부터 완료할 때까지 전 과정을 빠짐없이 순서대로 설명
3. 설치 매뉴얼에는 목차, 개요, 기본사항 등이 기본적으로 포함
4. ~~사용자 기준 작성~~
5. ~~목차에는 전체 설치 과정을 순서대로 요약한 후 관련 내용의 시작 페이지를 함께 기술~~
6. ~~개요에는 설치 매뉴얼의 주요 특징, 구성, 설치 방법, 순서 등의 내용을 기술~~

기본 사항

1. 소프트웨어 개요
2. 설치 관련 파일
3. 프로그램 삭제
4. ~~설치 아이콘~~
5. ~~관련 추가 정보~~

B 2-3-5 소프트웨어 사용자 매뉴얼 작성

사용자 매뉴얼 작성절차

1. 작성 지침 정의
2. 사용자 매뉴얼 구성 요소 정의(사용 설명서 검토)
3. 구성 요소별 내용 작성
4. 사용자 매뉴얼(설명서) 검토

A 2-3-6 소프트웨어 버전 등록

소프트웨어 패키징의 형상관리

1. 소프트웨어에 일어나는 수정이나 변경을 알아내고 제어
2. 목적 : 소프트웨어 개발의 전체 비용을 줄이고 개발 과정의 여러 방해 요인을 최소화되도록 보증
3. 관리 항목 : 소스 코드, 계획서, 요구 분석서, 운영 및 설치 지침서, 설계서, 프로그램, 테스트 케이스
4. 형상관리 : 소프트웨어에 가해지는 변경을 제어하고 관리
5. 형상관리 : 유지 보수 뿐 아니라 개발 단계에도 적용

형상관리의 중요성

1. 동일한 프로젝트에 대해 여러 개발자 동시 개발 가능
2. 소스 수정 제한
3. 배포본 관리
4. ~~진행 정도를 확인~~
5. ~~버그, 수정사항 추적 가능~~
6. ~~변경 사항을 체계적으로 추적하고 통제~~

형상 관리 기능 : 가시성과 추적성을 보장함으로써 소프트웨어의 생산성과 품질을 높일 수 있다.

1. 버전 제어
2. 형상 식별 : 형상 관리 계획을 근거로 형상관리의 대상이 무엇인지 식별하는 과정이다.
3. ~~형상 통제 (변경 관리)~~
4. 형상 검사 : 계획대로 형상관리가 진행되고 있는지, 요구 사항에 맞도록 제대로 이뤄졌는지 등을 살핀다
5. ~~형상 기록 (상태 보고)~~

소프트웨어의 버전 등록 관련 주요 기능 – 동체가 커저(동기화, 체크, 가져오기, 커밋, 저장소)

1. 체크인 : 저장소의 파일을 새로운 버전으로 갱신
2. 체크아웃
3. 커밋
4. ~~저장소, 가져오기, 동기화~~

B 2-3-7 소프트웨어 버전 관리 도구

공유 폴더 방식

1. 종류 : SCCS, PVCS, QVCS
    A. RCS :
        i. 동시에 소스를 수정하는 것을 방지,
        ii. 다른 방향으로 개발 결과를 합치거나 변경 내용을 추적할 수 있는 버전관리 도구

분산 저장소 방식

1. 버전 관리 자료가 원격 저장소와 로컬 저장소에 함께 저장되어 관리
2. 로컬 저장소에 버전 관리가 가능하므로 원격 저장소에 문제가 생겨도 로컬 저장소에서 작업
3. 종류 : Git , ~~GNU arch, DCVS, Bazaar, Mercurial, TeamWare, Bitkeeper, Plastic SCM~~

B 2-3-8 빌드 자동화 도구

개념

1. 빌드 자동화 도구 :
    A. 지속적인 통합 개발 환경에서 유용
    B. 종류 : Ant, Gradle, Jenkins, Maven 등

Gradle

1. 실행할 처리 명령들을 모아 태스크로 만든 후 태스크 단위로 실행
2. ~~빌드 속도를 향상~~
3. ~~JAVA, C, C++, Python 빌드 가능~~
4. ~~DSL을 스크립트 언어로 사용~~

2-4

B 2-4-1 애플리케이션 테스트

기본 원리

1. 결함 집중 :
    A. 파레토 법칙이 좌우
    B. 애플리케이션 결함의 대부분은 소수의 특정한 모듈에 집중되어 존재
    C. 결함은 발생한 모듈에서 계속 추가로 발생할 가능성이 높다.

검증과 확인

1. 결함 유형을 명확하게 하는데 도움
2. 검증 : 작업 제품이 요구 명세의 기능, 비기능 요구사항을 얼마나 잘 준수하는지 측정
3. 검증 : 개발 과정을 테스트 / 개발자의 입장
4. 확인 : 결과를 테스트 / 사용자의 입장

Pareto(파레토) 법칙

1. 소프트웨어 테스트에서 오류 80%는 전체 모듈의 20% 내에서 발견된다.

B 2-4-2 애플리케이션 테스트의 분류

목적에 따른 테스트

1. 강도 : 시스템에 과다 정보량을 부과하여 과부하 시에도 시스템이 정상적으로 작동되는지를 테스트
2. ~~회복 : 시스템에 고의로 실패를 유도하고 시스템이정상적으로 복귀하는지 테스트한다.~~
3. ~~안전 : 부당하고 불법적인 침입을 시도하여 보안시스템이 불법적인 침투를 잘 막아내는지 테스트한다.~~
4. ~~성능 : 시스템이 응답하는 시간 내에 처리하는 업무량, 사용자 요구에 시스템이 반응하는 속도 등~~
5. ~~구조 , 회귀 , 병행 등 50가지 이상~~

A 2-4-3 테스트 기법에 따른 애플리케이션 테스트

단어

1. 테스트 케이스 : 일반적으로 시험 조건, 테스트 데이터, 예상 결과가 포함
2. 기본 경로 : 수행 가능한 모든 경로를 의미
3. 테스트 데이터 :
    A. 동적 테스트 – 화이트박스 테스트 등
    B. 검증 기준(Test Coverage)을 정한다

화이트 박스 테스트 (White Box Testing)

1. 모듈의 논리적인 구조를 체계적으로 점검
2. Source Code의 모든 문장을 한 번 이상 수행함으로써 진행
3. 모듈 안의 작동을 직접 관찰
4. 산출물의 각 기능별로 적절한 프로그램의 제어 구조에 따라 선택, 반복 등의 부분을 수행
5. 논리 흐름도를 이용할 수 있다

화이트 박스 테스트 종류

1. Base Path Testing(기초 경로 검사)
2. Control Structure Testing(제어 구조 검사) :
    A. 데이터 흐름 검사
    B. 루프 검사
    ~~C. 조건 검사~~

블랙박스 테스트

1. 프로그램의 구조를 고려하지 않는다

블랙박스 테스트 종류

1. Equivalence Partitioning Testing (동치 분할 검사) :
2. Boundary Value Analysis (경계값 분석) :
3. Cause-Effect Graphing (원인-효과 그래프 검사) :

4. Error Guessing (오류 예측 검사)

~~5. Comparison Testing (비교 검사)~~

A 2-4-4 개발 단계에 따른 애플리케이션 테스트

단위 테스트

1. 개별 모듈을 시험하는 것 (모듈이 정확하게 구현, 예정한 기능이 제대로 수행되는지 점검)
2. 오류
    A. 알고리즘 오류에 따른 원치 않는 결과
    B. 탈출구가 없는 반복문의 사용
    C. 틀린 계산 수식에 의한 잘못된 결과

인수 테스트

1. 알파 테스트 :
    A. 개발자의 장소에서 사용자가 개발자 앞에서 행하는 기법
    B. 오류와 사용상의 문제점을 사용자와 개발자가 함께 확인
2. 베타 테스트 :
    A. 필드 테스팅
    B. 개발자 없이 고객의 사용 환경에 소프트웨어를 설치하여 검사를 수행하는 기법

~~3. 사용자 인수 테스트 :~~

~~4. 운영상의 인수 테스트 :~~

~~5. 계약 인수 테스트 :~~

~~6. 규정 인수 테스트 :~~

A 2-4-5 통합 테스트

개요

1. 모듈의 인터페이스와 결합을 테스트
2. 동작이 정상적으로 잘되고 있는지를 빨리 파악하고자 할 때, 하향식 통합 테스트를 사용하는 것이 좋다.

하향식

1. 넓이 우선 방식, 깊이 우선 방식
2. 모듈 간의 인터페이스와 시스템의 동작이 정상적인지 파악
3. 상위 컴포넌트를 하고 점증적으로 하위 컴포넌트를 테스트
4. 레벨이 낮은 데이터 구조의 세부 사항은 설기 초기 단계에서 필요
5. 통합 검사 시 인터페이스가 이미 정의되어 있어 통합이 간단
6. 하위 컴포넌트를 개발이 완료되지 않은 경우 스텁을 사용
7. Stub(스텁, 스터브) :
    A. 일시적으로 필요한 조건만을 가지고 임시로 제공되는 시험용 모듈

상향식

1. 하위 컴포넌트를 설계하고 완성되면 상위 컴포넌트를 결합하여 테스트
2. 테스트 드라이버 :
    A. 필요에 따라 매개 변수를 전달하고 모듈을 수행한 후의 결과를 보여줄 수 있다.
    B. 시험 대상 모듈을 호출하는 간이 소프트웨어
3. 단위 테스트에서 테스트의 대상이 되는 하위 모듈을 호출하고 파라미터를 전달하는 가상의 모듈

B 2-4-7 테스트 케이스/테스트 시나리오/테스트 오라클

테스트 케이스

1. 프로그램에 결함이 있더라도 입력에 대해 정상적인 결과를 낼 수 있는 테스트 케이스 찾기
2. 개발된 서비스가 정의된 요구사항을 준수하는지 확인하기 위한 입력 값, 실행 조건, 예상 결과의 집합

테스트 오라클

1. 참, 거짓 값을 입력하여 비교하는 기법 및 활동
2. 테스트 케이스 실행이 통과되었는지 실패하였는지 판단하기 위한 기준

테스트 오라클 종류

1. 참 :
2. 샘플링 : 특정한 몇몇 테스트 케이스의 입력 값들에 대해서만 기대하는 결과를 제공
3. 추정(휴리스틱, Heuristic) :
    A. 특정 테스트 케이스의 입력 값에 대해 기대하는 결과를 제공
    B. 나머지 입력 값들에 대해서는 추정으로 처리
4. 일관성 검사 : 애플리케이션의 변경이 있을 경우 테스트 케이스의 전과 후의 결과 값이 동일한지 확인

B 2-4-8 테스트 자동화 도구

도구 유형

1. 테스트 케이스 자동 생성 도구
    A. 입력 도메인 분석
    B. 랜덤 테스트
    C. 자료 흐름도
    ~~D. 기능 테스트~~
2. 성능 테스트 도구 : 성능 목표의 달성 여부를 확인 (처리량, 응답 시간, 경과 시간, 자원 사용률)
3. ~~정적 분석 도구, 테스트 실행 도구, 테스트 통제 도구, 테스트 하네스 도구~~

B 2-4-9 결함 관리

Fault (결함) 정의

소프트웨어 개발 활동을 수행함에 있어서 시스템이 고장을 일으키게 하여 오류가 있는 경우 발생

결함 = 고장, 오류, 오작동 등

A 2-4-11 복잡도

빅오 표기법

1. O(1) : 알고리즘 수행시간이 입력 데이터 수와 관계 없이 일정
2. $O(n\log_2 n)$: 2-Way 합병 정렬 , 힙 정렬(Heap Sort)
3. ~~$O(\log_2 n)$~~
4. ~~$O(n)$~~ : 이진 탐색트리 : 검색 효율이 가장 나쁜트리
5. ~~$O(n^2)$: 선택 정렬~~
6. ~~$O(2^n)$~~

순환 복잡도

1. 순환(Cyclomatic) : V(G) = E – N + 2 : E는 화살표 수, N은 노드의 수

힙 정렬

1. 정렬할 입력 레코드들로 힙을 구성
2. 가장 큰 키 값을 갖는 루트 노드를 제거하는 과정을 반복하여 정렬
3. 완전 이진트리로 입력자료의 레코드를 구성
4. 최적, 평균, 최악 수행 시간 모두 $O(n\log_2 n)$이다

A 2-4-12 애플리케이션 성능 개선

21.08.14 2장) 코드의 간결성을 유지하기 위해 사용되는 지침

1. 공백을 이용하여 실행문 그룹과 주석을 명확히 구분한다.
2. 복잡한 논리식과 산술식은 괄호와 들여쓰기(Indentation)를 통해 명확히 표현한다.
3. 빈 줄을 사용하여 선언부와 구현부를 구별한다.

21.08.14 2장) 품질 목표 항목

1. Portability 이식성 : 하나 이상의 하드웨어 환경에서 운용되기 위해 쉽게 수정될 수 있는 시스템 능력
2. Efficiency 효율성: 최소의 작업으로 요구되는 기능을 수행하는 정도
3. Usability 사용 용이성: 소프트웨어를 쉽게 사용할 수 있는 정도
4. Correctness 정확성: 사용자의 요구사항을 충족시키는 정도

소스 코드 최적화

1. 클린코드 작성원칙
    A. 가독성 : 누구나 코드를 쉽게 읽을 수 있도록 작성
    B. 단순성
        i. 한 번에 한가지 처리만 수행
        ii. 클래스/메소드/함수를 최소 단위로 분리
    C. 의존성 배제 : 다른 모듈에 미치는 영향을 최소화
    D. 중복성 최소화
    E. 추상화
2. 클린코드
    A. 누구든지 쉽게 이해하는 코드 작성
    B. 다른 모듈에 미치는 영향 최소화
    C. 단순, 명료한 코드 작성
3. 나쁜 코드
    A. Alien 외계인 코드 : 아주 오래되거나 참고문서 또는 개발자가 없어 유지 보수 작업이 어려운 코드

B. 스파게티 코드

소스 코드 품질 분석 도구

1. 정적 분석 도구
   A. 소스 코드를 실행시키지 않고 분석
   B. 코드에 있는 오류나 잠재적인 오류를 찾아내기 위한 활동
   C. 자료 흐름이나 논리 흐름을 분석하여 비정상적인 패턴을 찾기
2. 정적 분석 도구 종류 : pmd, cppcheck, checkstyle, ~~SonarQube , ccm, cobertura~~
3. ~~동적 분석 도구 종류 : Avalanche, Valgrind~~

2-5 인터페이스 구현

B 2-5-2 모듈 연계를 위한 인터페이스 기능 식별

개요

1. EAI :
   A. Point-to-Point
   B. Hub & Spoke
   C. Message Bus
   D. Hybrid :
      i. Hub & Spoke + Message Bus 혼합 방식
      ii. 필요한 경우 한 가지 방식으로 EAI 구현 가능
      iii. 데이터 병목 현상을 최소화
2. FEP

   입력되는 데이터를 컴퓨터의 프로세서가 처리하기 전에 미리 처리하여

   프로세서가 처리하는 시간을 줄여주는 프로그램이나 하드웨어를 말하는 것

B 2-5-5 인터페이스 구현

종류 : JSON, XML, AJAX, YAML, ~~REST~~

JSON :

1. 용량이 적은 데이터를 교환하기 위해 데이터 객체를 속성-값의 쌍 형태로 표현하는 형식
2. 자바 스크립트를 토대로 개발되어진 형식

AJAX :

1. 자바 스크립트를 사용한 비동기 통신 기술
2. 클라이언트와 서버간의 XML 데이터를 주고 받는 기술

A 2-5-7 인터페이스 보안

인터페이스 보안 기능 적용

1. 네트워크 영역
    A. 방식 : IPSec , SSL , S-HTTP
2. ~~애플리케이션 영역~~
3. ~~데이터베이스 영역~~

데이터 무결성 검사 도구

1. Tripwire : 크래커가 침입하여 백도어를 만들어 놓거나 설정 파일을 변경했을 때 분석하는 도구

A 2-5-9 인터페이스 구현 검증

인터페이스 구현 검증 도구

1. xUnit 단위 테스트 도구 : JUnit , CppUnit , HttpUnit , ~~NUnit~~
2. STAF :
    A. 서비스 호출, 컴포넌트 재사용 등 *다양한 환경*을 지원하는 테스트 프레임워크
    B. 각 테스트 대상 분산 환경에 *데몬을 사용*하여 테스트 대상 프로그램을 통해 테스트를 수행
    C. 통합하여 자동화 하는 검증 도구
3. NTAF : STAF의 장점인 재사용 및 확장성을 통합한 네이버의 테스트 자동화 프레임워크이다.
4. Water

~~5. FitNesse : 웹 기반 테스트케이스 설계, 실행, 결과 확인 등을 지원하는 테스트 프레임워크~~

~~6. Selenium~~

3과목 데이터베이스 구축

3-1 논리 데이터베이스 설계

A 3-1-1 데이터베이스 설계

데이터베이스 설계 순서 – 요구>개념적>논리적>물리적>구현

1. 요구 조건 분석 : ~~요구 조건 명세서 작성~~
2. 개념적 설계 : 개념 스키마, 트랜잭션 모델링, E-R 모델
3. 논리적 설계 : 스키마의 평가 및 정제
4. 물리적 설계 : 응답 시간, 저장 공간의 효율화, 트랜잭션 처리량
5. 구현 : ~~목표 DBMS의 DDL(정의어)로 데이터베이스 생성, 트랜잭션 작성~~

개념적 설계

1. 산출물로 E-R Diagram을 만들 수 있다.
2. DBMS에 독립적인 개념 스키마를 설계

논리적 설계 (데이터 모델링)

1. 논리적 데이터베이스 구조로 매핑
2. 트랜잭션 인터페이스 설계
3. 스키마의 평가, 정제
4. 테이블을 설계하는 단계

물리적 설계 (데이터 구조화)

1. 물리적 설계의 목적은 효율적인 방법으로 데이터를 저장하는 것
2. 트랜잭션 처리량, 응답시간, 디스크 용량, 저장공간 등을 고려
3. 저장 레코드의 형식, 순서, 접근 경로와 같은 정보를 사용하여 설계

저장 레코드의 양식 고려 사항

1. 데이터 타입
2. 데이터 값의 분포
3. 접근 빈도

B 3-1-2 데이터 모델의 개념

데이터 모델에 표시할 요소

1. 논리적 데이터 구조
2. 연산 (Operation) : 개념 설계나 컴퓨터 설계에서 실제로 표현된 값들을 처리하는 작업
3. 제약 조건

A 3-1-7 E-R(개체-관계) 모델

개요 – 정보공학 방법론

1. 특정 DBMS를 고려하여 제작하지 않는다
2. 개념적 데이터베이스 단계에서 제작
3. E-R 모델의 기본적인 아이디어를 시작적으로 가장 잘 나타냄
4. ~~개념적 논리 데이터로 표현하기 위한 방법~~
5. ~~개체, 관계 속성으로 묘사~~
6. ~~1:1, 1:N, N:M 등의 관계 유형을 제한 없이 나타냄~~
7. ~~나중에는 일반화 계층 같은 복잡한 개념들이 첨가되어 확장된 모델로 발전했다~~

E-R 다이어그램

1. 선, 링크 : 개체 타입과 속성을 연결
2. 사각형 : 개체 타입
3. 마름모 : 관계 타입
4. 타원 : 속성
5. 이중 타원 : 다중값 속성(복합 속성)
6. ~~밑줄 타원 : 기본키 속성~~
7. ~~복수 타원 : 복합 속성~~
8. ~~관계 : 1:1, 1:N, N:M 등의 개체 간의 관계에 대한 대응수를 선 위에 기술~~

A 3-1-9 관계형 데이터베이스의 구조

관계형 데이터베이스의 Relation 구조

1. 튜플(행) :
    A. Cardinality 카디널리티 : 튜플의 수
    ~~B. 레코드와 같은 의미~~
2. 속성(열, 차수) :
    A. 개체의 특성을 기술
    B. 데이터베이스를 구성하는 가장 작은 논리적 단위
    C. 파일 구조상 데이터 항목, 데이터 필드에 해당
    ~~D. 디그리 : 속성의 수~~
3. 도메인 :
    A. 애트리뷰트(속성)가 가질 수 있는 원자값들의 집합
    B. 애트리뷰트(속성) 값이 나타날 때 합법 여부를 시스템이 검사하는데 사용

릴레이션(테이블)의 특징

1. 모든 속성 값은 원자 값을 갖는다
2. 릴레이션의 각 행을 스키마라 한다 (예 : 도서번호, 도서명, 저자, 가격 등)
3. 릴레이션의 각 열을 튜플이라 한다.
4. 한 릴레이션에 포함된 튜플은 모두 상이하다
5. 한 릴레이션에 포함된 튜플 사이에는 순서가 없다
6. 한 릴레이션을 구성하는 속성 사이에는 순서가 없다
7. 한 릴레이션에는 동일한 이름의 속성이 있을 수 없다
8. 속성을 구성하는 값에는 동일한 값이 있을 수 있다
~~9. 튜플들의 삽입, 삭제 등의 작업으로 인해 릴레이션은 시간에 따라 변한다~~
~~10. 튜플을 유일하게 식별하기 위해 속성들의 부분집합을 키(Key)로 설정~~

A 3-1-10 관계형 데이터베이스의 구조 제약조건 - Key

Candidate 후보키

1. 개체들을 고유하게 식별할 수 있는 속성
2. 모든 튜플에 대해 유일성과 최소성을 모두 만족
3. ~~중복된 튜플은 없다~~
4. ~~반드시 하나 이상의 후보키가 존재한다~~

기본키

1. NOT NULL 값을 가지지 않는다
2. 릴레이션에서 튜플을 유일하게 구별할 수 있다
3. 외래키로 참조될 수 있다
4. ~~후보키의 성질을 갖는다 / 유일성과 최소성을 가진다~~

대체키 (보조키)

1. 기본키를 제외한 나머지를 후보키라고 부른다

슈퍼키

1. 한 개 이상의 속성들의 집합으로 구성된 키
2. 유일성은 만족시키지만 최소성은 만족시키지 못함

외래키

1. 다른 릴레이션의 기본키를 참조
2. 다른 테이블의 기본키로 사용되는 속성

A 3-1-11 관계형 데이터베이스의 구조 제약조건–무결성

데이터베이스 무결성 규정

1. 데이터가 만족해야 될 제약 조건, 규정을 참조할 때 사용하는 식별자 등의 요소가 포함될 수 있다.
2. 무결성 규정의 대상으로는 도메인, 키(개체), 종속성(참조) 등이 있다.

릴레이션 무결성 규정

1. 릴레이 션을 조작하는 과정에서의 의미적 관계을 명세한 것이다.

개체 무결성 (키본키 관련)

1. 키본키에 속해 있는 애트리뷰트(속성)는 NULL값이나 중복 값을 가질 수 없다

도메인 무결성

1. ~~특정 속성의 값이 그 속성이 정의된 도메인에 속한 값이어야 한다는 규정~~

참조 무결성 (외래키 관련)

1. NULL 가능하다
2. 키본키 값과 동일해야 한다

A 3-1-12 관계 대수 및 관계해석

개요

1. 순수 관계 연산자 : Select 셀렉트 σ, Project 프로젝트 π, Join 조인 ⋈, Division 디비전 ⊂
2. ~~일반 집합 연산자~~ : Cartesian Product (교차곱) ~~Union (합집합) Intersection (교집합) Difference (차집합)~~
3. 연산의 집합
4. 수행해야 할 연산의 순서를 명시

관계 대수

1. 절차적 언어
2. 릴레이션 조작을 위한 연산의 집합으로 피연산자와 결과가 모두 릴레이션이다.
3. 일반 집합 연산과 순수 관계 연산으로 구분된다.
4. 질의에 대한 해를 구하기 위해 수행해야 할 연산의 순서를 명시한다.

~~관계해석~~

1. ~~비절차적 언어~~

일반 집합 연산자

1. ~~Cartesian Product (교차곱)~~ ×
2. ~~Union (합집합)~~ ∪
3. ~~Intersection (교집합)~~ ∩
4. ~~Difference (차집합)~~ −

관계해석

1. ∀ : 모든 것에 대하여
2. ~~∃ : 하나라도 존재한다~~

A 3-1-13 정규화

개요

1. 1NF, 2NF, 3NF, BCNF, 4NF, 5NF 등 있다.

목적

1. 데이터베이스 내에서 표현 가능하게 만든다
2. 데이터 삽입 시 릴레이션을 재구성할 필요성을 줄인다
3. 효과적인 검색 알고리즘을 생성
4. 중복을 배제하여 이상(Anomaly)의 발생을 방지
5. 데이터 구조의 안전성 최대화
6. 수정 삭제 시 이상 현상의 최소화
7. 테이블 불일치 위험의 최소화
8. 중복을 배제
9. 삽입, 삭제, 갱신 이상(Anomaly)의 발생을 방지

이상(Anomaly)의 개념 및 종류

1. 삽입 이상 :

   릴레이션에서 데이터를 삽입할 때 의도와는 상관없이 원하지 않는 값들도 함께 삽입되는 현상

2. 삭제 이상 :

   릴레이션에서 한 튜플을 삭제할 때 의도와는 상관없는 값들도 함께 삭제되는 연쇄 삭제 현상

3. 갱신 이상 :

   릴레이션에서 튜플에 있는 속성값을 갱신할 때 일부 튜플의 정보만 갱신되어 정보에 모순이 생기는 현상

정규화 과정 – 도부이결다조

1. 1NF : 도메인이 원자값
2. 2NF : 부분적 함수 종속 제거 / 완전 함수적 종속 관계를 만족
3. 3NF : 이행적 함수 종속 제거 / A->B , B->C , A->C
4. BCNF : 결정자가 후보키가 아닌 함수 종속 제거 / 결정자 = 후보키

5. 4NF : 다치 종속 제거
6. 5NF : 조인 종속성 이용

B 3-1-14 반정규화

개념

1. 시스템의 성능 향상과 개발 운영의 단순화를 위해 중복, 통합, 분리 등을 수행하는 데이터 모델링 기법

중복 테이블 추가

1. 집계 테이블 추가
2. 진행 테이블 추가
3. 특정 부분만을 포함하는 테이블의 추가

B 3-1-10 시스템 카탈로그

의미

1. 시스템 자체에 관련 있는 다양한 객체에 관한 정보를 포함하는 시스템 베이스
2. 기본 테이블, 뷰, 인덱스, 패키지, 접근 권한 등의 정보를 저장

저장 정보

1. 메타 데이터 = 시스템 카탈로그에 저장된 정보

특징

1. 카탈로그의 갱신 : DBMS가 스스로 생성하고 유지

데이터 사전 = 시스템 카탈로그 = 시스템 데이터베이스

1. 메타 데이터를 저장하고 있다
2. 데이터 사전에 있는 데이터에 접근하기 위한 위치 정보는 데이터 디렉토리에서 관리

3-2 물리 데이터베이스 설계

A 3-2-3 트랜잭션 분석 / CRUD 분석

트랜잭션 정의

1. 데이터베이스에서 하나의 논리적 기능을 수행하기 위한 작업의 단위
2. 한꺼번에 모두 수행되어야 할 일련의 연산들을 의미

트랜잭션 상태

1. Partially Committed(부분 완료) : 마지막 연산이 실행된 직후의 상태, 최종 결과를 반영하지 않은 상태
2. Isolation(독립성, 격리성, 순차성) :

    둘 이상의 트랜잭션이 동시에 병행 실행되는 경우,

    어느 하나의 트랜잭션 실행 중에 다른 트랜잭션의 연산이 끼어들 수 없음
3. Consistency(일관성) : 시스템이 가지고 있는 고정요소는 트랜잭션 수행 전 = 트랜잭션 수행 완료 후
4. ~~Durability(영속성, 지속성)~~

트랜잭션 특성

1. Atomicity (원자성) :
    A. 트랜잭션의 연산은 모두 실행되거나, 모두 실행되지 않아야 한다
    B. Commit과 Rollback 명령어에 의해 보장 받는다

CRUD 분석

1. 생성, 읽기, 갱신, 삭제 연산으로 프로세스와 테이블 간에 매트릭스를 만들어 트랜잭션을 분석

B 3-2-4 인덱스 설계

개념

1. 자료를 빠르게 조회하기 위하여 사용 / 검색 시 처리 속도 향상 / 검색 성능을 최적화
2. 테이블을 삭제하면 인덱스도 같이 삭제
3. B-트리 인덱스 = Branch Block(분기목적) , Leaf Block 있다
4. BETWEEN 등 범위(Range) 검색에 활용

A 3-2-5 뷰 설계

뷰 특징

1. 가상 테이블
2. 논리적으로 존재
3. 카탈로그에 저장
4. CREATE문을 사용하여 정의
5. 제거할 때 DROP문 사용
6. ALTER문을 이용하여 변경
7. UPDATE에는 제약이 있다
8. 뷰 위에 또 다른 뷰를 정의할 수 있다.
9. 다른 뷰를 기반으로 새로운 뷰를 만들 수 있다
10. 삽입, 갱신, 삭제 연산은 제약이 따른다
11. 뷰가 정의된 기본 테이블이 제거되면 뷰도 자동적으로 제거된다.
12. DBA는 보안 측면에서 뷰를 활용
13. 독립적인 인덱스를 가질 수 없다

뷰의 장단점

1. 장점
    A. 논리적 독립성을 제공
    B. 데이터 보안 용이
    C. 사용자 데이터 관리 용이
    ~~D. 접근 제어를 통한 자동 보안이 제공~~
2. 단점
    A. 독립적인 인덱스를 가질 수 없다
    ~~B. 뷰의 정의를 변경할 수 없다~~
    ~~C. 삽입, 삭제, 갱신 연산에 제약이 따른다~~

B 3-2-7 파티션 설계

파티션 종류

1. 범위 분할(Range) : 지정한 열의 값을 기준으로 범위를 지정하여 분할
2. 해시 분할(Hash)
3. 조합 분할(Composite)
4. 라운드 로빈 분할(Round Robin)
5. ~~목록 분할(List)~~

B 3-2-9 분산 데이터베이스 설계

분산 데이터베이스 정의

1. 물리적으로 분산된 데이터베이스 시스템을 논리적으로 하나의 데이터베이스 시스템처럼 사용
2. 통신 네트워크 구조가 데이터 통신에 영향을 주므로 효율적으로 설계해야 함

구성요소

1. 분산 처리기 : 지역 컴퓨터
    A. 물리적으로 분산되어 지역별로 필요한 데이터를 처리할 수 있는 지역 컴퓨터

목표

1. Location Transparency (위치 투명성)
2. Replication Transparency (중복 투명성)
3. Concurrency Transparency (병행 투명성)
4. Failure Transparency (장애 투명성) : 장애가 발생해도 데이터 무결성 보장

장단점

1. 장점
    A. 지역 서버의 고유 데이터에 대한 작업은 중앙 서버의 통제 없이 자유롭게 수행
    B. 새로운 지역 서버를 추가하거나 장비를 추가하는 등의 작업이 용이
    C. 논리적으로는 하나의 시스템에 속하지만 물리적으로 여러 개의 컴퓨터 사이트에 분산
2. 단점
    A. 설계 어렵다
    B. 개발 비용 증가
    C. 처리 비용 증가

B 3-2-11 데이터베이스 보안/암호화

암호화

1. 평문 : 암호화되기 전의 원본 메시지
2. 암호문 : 암호화가 적용된 메시지
3. 키(Key) : 적절한 암호화를 위하여 사용하는 값
4. 암호화 : 평문을 암호문으로 바꾸는 작업
5. 복호화 : 암호문을 평문으로 바꾸는 작업

A 3-2-12 데이터베이스 보안-접근통제

접근 통제

1. DAC(Discretionary Access Control) 임의 :
2. MAC(Mandatory Access Control) 강제 :
    A. 높은 보안 수준을 요구하는 정보(객체)가 낮은 보안 수준의 주체에게 노출되지 않도록 함
3. RBAC(Role Based Access Control) 역할기반 : 조직 내에서 직무, 직책 등 개인의 역할에 따라 부여

강제 접근통제(MAC)의 보안 모델

1. Bell-LaPadula Model :
    A. 군대의 보안처럼 정보의 기밀성에 따라 상하 관계가 구분된 정보를 보호하기 위해 사용
    B. 자신의 권한보다 보안 레벨이 높은 것은 읽을 수 없고, 낮은 것은 읽기 가능
    C. 자신의 권한보다 보안 레벨이 높은 것은 쓰기 가능, 낮은 것은 쓰기 제한
2. Biba Integrity Model
3. Clark-Wilson Integrity Model
4. Chinese Wall Model

B 3-2-14 스토리지

DAS

1. 하드디스크와 같은 데이터 저장장치를 호스트버스 어댑터에 직접 연결
2. 저장장치와 호스트 기기 사이에 네트워크 디바이스가 있지 말아야 함
3. 직접 연결하는 방식

SAN

1. 네트워크상에 광채널 스위치의 이점인 고속 전송과 장거리 연결 및 멀티 프로토콜 기능을 활용
2. 네트워크상에 동일 저장장치의 데이터를 공유하게 함
3. 여러 개의 저장장치나 백업 장비를 단일화시킨 시스템

3-3 SQL 응용

A 3-3-1 SQL의 개념

SQL의 분류

1. DDL : 정의
    A. CREATE : 스키마, 도메인, 인덱스, 테이블, 뷰 정의
    B. ALTER : 변경
    C. DROP : 삭제
2. DML : 조작
    A. SELECT
    B. INSERT
    C. DELETE
    D. UPDATE
3. DCL : 제어 : 데이터 보안, 무결성 유지, 병행수행 제어
    A. COMMIT
    B. ROLLBACK
    C. GRANT
    D. REVOKE

A 3-3-2 DDL

CREATE TABLE TO

1. 속성의 NOT NULL 여부 지정
2. 기본키를 구성하는 속성 지정
3. CHECK 제약조건 정의

DROP : 스키마, 도메인, 뷰, 인덱스 삭제

1. CASCADE : 참조하는 개체를 함께 제거

A 3-3-3 DCL

개념

1. 데이터 관리(제어) 목적

GRANT / REVOKE :

1. GRANT : 권한 부여
    A. 권한리스트 ON 개체 TO 사용자
2. REVOKE : 권한 취소
    A. 권한리스트 ON 개체 FROM 사용자
    B. 데이터베이스 조작 작업이 비정상적으로 종료되었을 때 원래 상태로 복구한다.

COMMIT : 반영 완료

ROLLBACK : 취소 복구, 철회(Aborted)

A 3-3-4 DML

삭제문

1. DELETE FROM~ [WHERE 조건]
2. 테이블의 행을 삭제할 때 사용

갱신문

1. UPDATE~ SET~ [WHERE 조건]

A 3-3-5 DML-SELECT-1

일반 형식

1. SELECT절
    A. PREDICATE
        i. ALL
        ii. DISTINCT : 중복된 튜플은 1개만 검색
        iii. DISTINCTROW
    B. 그룹함수(속성)
        i. COUNT : 그룹별 튜플 수
        ii. SUM : 그룹별 합계
        iii. AVG : 그룹별 평균
        iv. MAX : 그룹별 최대값
        v. MIN : 그룹별 최소값
        vi. STDDEV : 그룹별 표준편차
        vii. VARIANCE : 그룹별 분산
        viii. ROLLUP : 그룹별 소계 (인수로 주어진 속성을 대상)
        ix. CUBE : 그룹별 소계 (인수로 주어진 속성을 대상으로 모든 조합)
    C. WINDOW 함수 : GROUP BY절을 사용하지 않고 함수의 인수로 지정한 속성을 범위로 지정
        i. ROW_NUMBER() : 레코드에 대한 일련 번호 반환
        ii. RANK() : 공동 순위 반영
        iii. DENSE_RANK() : 공동 순위를 무시하고 순위 부여
2. FROM절
3. WHERE절
    A. GROUP BY절 : 속성을 기준으로 자료를 그룹화
    B. HAVING절 : GROUP BY가 반드시 사용된 후 적용

4. ORDER BY절
    A. ASC : 오름차순
    B. DESC : 내림차순

연산자

1. % : 모든 문자를 대표
2. _ : 문자 하나를 대표
3. # : 숫자 하나를 대표

기본 검색

1. SELECT~ FROM~ [WHERE 조건][ORDER BY]

조건 지정 검색

1. '튜플'
2. 속성 LIKE "김%" / 김씨 찾기
3. 생일 BETWEEN '01/01/70' AND #12/31/70# / 70년 1월 1일 ~ 70년 12월 31일 찾기
4. 주소 IS NULL / NULL관련일 때 IS 사용하기

하위 질의

1. SELECT~ FROM~ WHERE 속성 = (속성) / SELECT~ FROM~ WHERE 속성 IN (속성)
2. SELECT~ FROM~ WHERE 속성 != (속성) / SELECT~ FROM~ WHERE 속성 NOT IN (속성)

B 3-3-6 DML-SELECT-2

집합 연산자를 이용한 통합 질의

1. 2개 이상의 테이블의 데이터를 하나로 통합
2. UNION : 합집합 (중복된 행은 한 번만 출력)
3. UNION ALL : 합집합 (중복된 행은 그대로 출력)
4. INTERSECT : 교집합 (공통된 행만 출력)
5. EXCEPT : 차집합 (2번째 SELECT문의 조회 결과를 제외한 행을 출력)

~~B 3-3-7 DML-JOIN~~

INNER JOIN

1. EQUI JOIN
    A. SELECT~ FROM 테이블1, 테이블2 WHERE 테이블1.속성 = 테이블2.속성
    B. SELECT~ FROM 테이블1 NATURAL JOIN 테이블2
    C. SELECT~ FROM 테이블1 JOIN 테이블2 USING(속성)
2. NON-EQUI JOIN
    A. =가 아닌 비교 연산자를 사용(<, >, <=, >=)
    B. SELECT~ FROM 테이블1, 테이블2 WHERE (NON-EQUI JOIN 조건)

OUTER JOIN : INNER JOIN 결과 구한 후,

1. LEFT OUTER JOIN : 우측R의 어떤 튜플과 맞지 않은 좌측R 튜플에 NULL 붙임
    A. SELECT~ FROM 테이블1 LEFT OUTER JOIN 테이블2 ON 테이블1.속성 = 테이블2.속성
    B. SELECT~ FROM 테이블1, 테이블2 WHERE 테이블1.속성 = 테이블2.속성(+)
2. RIGHT OUER JOIN : 좌측R의 어떤 튜플과 맞지 않은 우측R 튜플에 NULL 붙임
    A. SELECT~ FROM 테이블1 RIGHT OUTER JOIN 테이블2 ON 테이블1.속성 = 테이블2.속성
    B. SELECT~ FROM 테이블1, 테이블2 WHERE 테이블1.속성(+) = 테이블2.속성

3-4 SQL 활용

B 3-4-2 트리거

개요

삽입, 갱신, 삭제 등의 이벤트가 발생할 때마다 관련 작업이 자동으로 수행

4과목 프로그래밍 언어 활용

4-1 서버 프로그램 구현

B 4-1-4 배치 프로그램

배치 프로그램 필수 요소

1. 대용량 데이터 : 대용량 데이터를 처리할 수 있어야 한다.
2. 자동화 : 심각한 오류 상황 외에는 사용자의 개입 없이 동작
3. 안전성/신뢰성 : 어떤 문제가 생겼는지, 언제 발생했는지 등을 추적할 수 있어야 한다
4. 성능 : 다른 응용 프로그램의 수행을 방해하지 않아야 함, 지정된 시간 내에 처리가 완료되어야 함
5. 견고성 : 잘못된 데이터나 데이터 중복 등의 상황으로 중단되는 일 없이 수행되어야 함

4-2 프로그래밍 언어 활용

B 4-2-1 데이터 타입

C/C++의 데이터 타입 크기 및 기억 범위

1. 정수 :
    A. Short
    B. Int
    C. Long
    D. Long Long

C언어의 구조체

1. Struct : 데이터를 처리할 때 사용
2. Strcmp : 연결

JAVA의 데이터 타입 크기 및 기억 범위

1. 변수 : 어떤 값을 주기억장치에 기억하기 위해서 사용하는 공간
2. 변수의 자료형에 따라 저장할 수 있는 값의 종류와 범위가 달라진다
3. Boolean 자료형 : 조건이 참, 거짓을 판단할 때 사용

Python의 시퀀스 자료형

1. 리스트 : 다양한 자료형의 값을 연속적으로 저장하여, 필요에 따라 개수를 늘리거나 줄일 수 있다
2. 튜플 : 리스트처럼 요소를 연속적으로 저장하지만, 요소의 추가, 삭제, 변경은 불가
3. Range : 연속된 숫자를 생성하는 것 (리스트, 방복문 등에 많이 사용)

A 4-2-2 변수

변수명 작성 규칙

1. 영문자, 숫자, _(under bar)를 사용
2. 첫 글자는 영문자, _(under bar)를 사용
3. 글자 수 제한 없음
4. 공백, +, -, /, * 등의 특수문자 사용 불가
5. 대소문자 구분
6. 예약어를 변수명으로 사용 불가
7. 변수 선언 시 문장 끝에 반드시 세미클론(;) 사용
8. 헝가리안 표기법 : 변수 선언 시 변수명에 테이터 타입을 명시하는 것을

Garbage Collector 가비지 콜렉터

JAVA에서 힙(Heap)에 남아있으나 변수가 가지고 있던

참조값을 잃거나 변수 자체가 없어짐으로써 더 이상 사용되지 않는 객체를

제거해주는 역할을 하는 모듈

A 4-2-3 연산자

산술 연산자

+, -, *, /, %, ++, --

비트 연산자

&, ^, |, ~

논리 연산자

!, &&, ||

조건 연산자

조건 ? 수식1 : 수식2

연산자 우선순위

1. 단항 연산자 : !, ~, ++, --, sizeof
2. 산술 연산자1 : *, /, %
3. 산술 연산자2 : +, -
4. 시프트 연산자 : <<. >>
5. 관계 연산자1 : <, >, <=, >=
6. 관계 연산자2 : ==, !=
7. 비트 연산자 : &, ^, |
8. 논리 연산자 : &&, ||
9. 조건 연산자 : ? :
10. 대입 연산자 : =, +=, -=, *=, /=, %=, <<=, >>=
11. 순서 연산자 : ,

B 4-2-4 데이터 입출력

JAVA에서 표준 출력

1. Print() , Println() ,Printf()

B 4-2-5 제어문

단순 if문

```
If(조건)
        실행할 문장 ;
```

다중 if문

```
If(조건1)
        실행할 문장1 ;
Else if(조건2)
        실행할 문장2 ;
Else
        실행할 문장3 ;
```

Switch문

```
Switch(수식)
Case 레이블1 :
        실행할 문장1 ;
        Break ;
Case 레이블2 :
        실행할 문장2 ;
        Break ;
Default ;
```

B 4-2-6 반복문

For문

For(초기값 ; 최종값 ; 증가값)

실행할 문장 ;

While문

While(조건)

실행할 문장 ;

Do~while문

Do

실행할 문장 ;

While(조건) ;

Break : swich문, 반복문에서 사용하며 블록을 벗어난다

Continue : 반복문 이후 문장을 실행하지 않고 처음으로 옮긴다

A 4-2-7 배열과 문자열

1차원 배열

자료형 변수명 [개수]

2차원 배열

자료형 변수명 [행개수][열개수]

배열의 초기화

1. 자료형 변수명 [개수] = {'A', 'B', 'C'}
2. 자료형 변수명 [] = {'A', 'B', 'C'}
3. 자료형 변수명 [행개수][열개수] = { {1, 2, 3} , {4, 5, 6} }
4. 자료형 변수명 [행개수][열개수] = { 1, 2, 3 , 4, 5, 6 }

B 4-2-8 포인터

포인터와 포인터 변수

1. 포인터 변수 : C언어에서 변수의 주소를 저장하는 변수
2. 포인터 변수에 주소를 저장할 때는 변수 앞에 &을 붙인다
3. 포인터 변수가 가리키는 곳은 *을 붙인다
4. 힙 영역에 접근하는 동적 변수

n = 4, pt = &n

&n = 변수 n의 주소값

*pt = pt가 가리키는 주소에 저장된 값 = n

포인터와 배열

자료형 변수명1 [개수] , *변수명2

변수명2 = 변수명1 -> 변수명1 [0]의 주소를 저장

변수명2 = &변수명1 [0] -> 변수명1 [0]의 주소를 저장

B 4-2-9 Python의 기초

데이터 입출력 함수

Input(입력) 함수 : 키보드로 입력받아 변수에 저장하는 함수

Split(분리문자) 함수 : input뒤에 사용하는 함수로 분리문자를 이용하여 분리한다

Print() 함수 : print(출력값1, 출력값2, … , sep = 분리문자, end = 종료문자)

Print(서식 문자열* %(출력값1, 출력값2, …))

% : 문자임을 지정

입력 값의 형변환

1. 변환 데이터 1개일 때
   A. 변수 = int(input())
   B. 변수 = float(input())
2. 변환 데이터 2개 이상일 때
   A. 변수1, 변수2 = map(int, input().split())
   B. 변수1, 변수2 = map(float, input().split())

리스트 : C / JAVA에서 배열

리스트명 = [값1, 값2, …]

리스트명 = list([값1, 값2, …])

딕셔너리

딕셔너리명 = {키1:값1, 키2:값2, …}

딕셔너리명 = dict{키1:값1, 키2:값2, …}

Range : 연속된 숫자를 생성

Range(최종값)

Range(초기값, 최종값)

Range(초기값, 최종값, 증가값)

튜플 : 시퀀스(Sequence) 데이터 타입에 해당, 순서에 따라 저장할 수 있으나 저장된 내용을 변경할 수 없다

슬라이브(SIice) : 문자열, 리스트와 같은 순차형 객체에서 일부 잘라 변환하는 기능

객체명[초기위치:최종위치]

객체명[초기위치:최종위치:증가값]

A 4-2-10 Python의 활용

If문

1. If 조건 :

   실행할 문장

2. If 조건 :

   실행할 문장1

   Else :

   실행할 문장2

3. if 조건1 :

   실행할 문장1

   Elif 조건2 :

   실행할 문장2

   Else :

   실행할 문장3

For문

1. for 변수 in range(최종값) :

   실행할 문장

2. for 변수 in 리스트 :

   실행할 문장

While문

1. while 참,거짓 조건 :

   실행할 문장

클래스

1. class 클래스명 :

   실행할 문장

   def 메소드명 (self, 인수) :

   실행할 문장

   return 값

2. 변수명 = 클래스명()
3. Self(전역변수) : 클래스에 속한 변수에 접근할 때 사용하는 명칭 (임의로 지정 가능)
4. 인수(매개변수, 파라미터) : 메소드에 변수 (임의으로 지정 가능)

B 4-2-13 스크립트 언어

종류

1. 자바스크립트 :
    A. 프로토타입의 개념이 존재
    B. Prototype Link와 Prototype Object를 활용
    C. 객체지향 언어
2. 쉘 스크립트 :
    A. 제어문 선택형 : if, case
    B. 제어문 반복형 : for, while, until
3. 파이썬 :
    A. 귀도 반 로섬이 발표한 언어
    B. 인터프리터 방식이자 객체지향적이며
    C. 배우기 쉽고 이식성이 좋다

~~4. VB스크립트~~

~~5. ASP~~

~~6. JSP~~

~~7. PHP~~

~~8. Basic~~

A 4-2-15 라이브러리

개념

1. 필요할 때 찾아서 쓸 수 있도록 모듈화 되어 제공되는 프로그램
2. 도움말, 설치파일, 샘플코드 등을 제공
3. 모듈과 패키지를 총칭
4. 모듈 : 개별파일
5. 패키지 : 파일들을 모아 놓은 폴더
6. 표준 라이브러리 : 프로그래밍 언어가 기본적으로 가지고 있음
7. 외부 라이브러리 : 별도의 파일 설치를 필요로 함

C언어의 대표적인 표준 라이브러리

1. Stdio.h : 데이터 입출력
2. Math.h : 수학 함수
3. String.h : 문자열 처리에 사용되는 기능
4. Stdlib.h : 자료형 변환, 난수 발생, 메모리 할당에 사용 / atoi(), atof(), srand(), rand(), malloc(), free() 등
5. Time.h : 시간 처리에 사용

4-3 응용 SW 기초 기술 활용

B 4-3-1 운영체제의 개념

구성

1. 감시 프로그램
2. 작업 관리 프로그램
3. 데이터 관리 프로그램

기능

1. 자원의 현재 상태 파악
2. 자원 분배를 위한 스케줄링을 담당
3. CPU, 메모리 공간, 기억장치, 입출력 장치, 계산기 등의 자원 관리
4. 입출력 장치와 사용자 프로그램 제어
5. ~~인터페이스 제공~~
6. ~~하드웨어, 네트워크 관리 제어~~

A 4-3-3 UNIX / LINUX / MasOS

UNIX 개요 및 특징

1. 하나 이상의 작업에 대하여 백그라운드에서 수행이 가능
2. Multi-User, Multi-Tasking 지원
3. 트리 구조의 파일 시스템
4. 이식성이 높다
5. 장치 간의 호환성이 높다
6. ~~C언어로 되어 있다~~
7. ~~크기가 작고 이해하기 쉽다~~
8. ~~전문적인 프로그램 개발이 용이~~
9. ~~다양한 유틸리티 프로그램들이 존재~~

UNIX 시스템의 구성

1. 커널 (자원관리)
    A. 프로세스 생성, 종류
    B. 기억장치 할당, 회수
    C. 파일 시스템 관리
    D. 프로세스, 메모리 관리
2. 쉘 (명령어 해석)
    A. 명령어 해석기
    B. 시스템과 사용자 간의 인터페이스를 담당
    C. 여러 종류의 쉘이 있다
    D. 사용자 명령을 해석하고 커널로 전달
    E. 반복적인 명령 프로그램을 만듦
    F. 초기화 파일을 이용해 사용자 환경을 설정
3. Utility Program

파일 디스크립터 (파일 제어 블록 이라고도 한다)

1. 파일 관리를 위해 시스템이 필요로 하는 정보를 가지고 있다.
2. 보조기억장치에 저장되어 있다가 파일이 개방(open)되면 주기억장치로 이동
3. ~~파일 시스템이 관리를 하므로 사용자가 직접 참조할 수 없다~~
4. ~~파일마다 독립적~~
5. ~~시스템에 따라 다른 구조~~

B 4-3-4 기억장치 관리의 개요

배치 전략

1. 최초 적합 (First) : 들어갈 수 있는 크기의 빈 영역 중에서 첫 번째에 배치
2. 최적 적합 (Best) : 들어갈 수 있는 크기의 빈 영역 중에서 단편화를 가장 적게 남기는 배치
3. 최악 적합 (Worst) : 들어갈 수 있는 크기의 빈 영역 중에서 단편화를 가장 많이 남기는 배치

A 4-3-6 가상기억장치 구현 기법/페이지 교체 알고리즘

Paging (페이징)기법 :

고정된 크기의 일정 블록으로 나눔

Segmentation (세그먼테이션) 기법 :

가변적인 크기의 블록을 나눔

페이지 교체 알고리즘

1. OPT : 최적 교체
2. FIFO : 선입 선출
3. LFU : 사용 빈도가 적은 페이지 교체
4. LRU : 오래된 페이지 교체
5. NUR : LRU와 비슷, 최근에 사용하지 않음
6. SCR : FIFO 단점 보안
7. Optimal

B 4-3-7 가상기억장치 기타 관리 사항

페이지 크기

1. 작을 경우
    A. 기억장소 이용 효율 증가
    B. 입출력 시간이 늘어남
    C. 내부 단편화 감소
    D. 맵 테이블 크기가 커짐
2. ~~클 경우~~

Locality

1. 시간 구역성
    A. 반복(루프), 스택, 부프로그램(서프루틴)
2. 공간 구역성
    A. 프로세스가 페이지를 참조했다면 이후

        가상주소공간상 그 페이지와 인접한 페이지들을 참조할 가능성이 높음을 의미

Working Set (워킹 셋)

1. 프로세스가 일정 시간동안 자주 참조하는 페이지들의 집합

스레싱

1. 프로세스 처리시간보다 페이지 교체에 소요되는 시간이 더 많아지는 현상

스레싱 현상 방지 방법

1. 프로세스가 필요로 하는 프레임을 제공
2. ~~다중 프로그래밍의 정도를 적정 수준으로 유지~~
3. ~~페이지 부제 빈도를 조절하여 사용~~
4. ~~워킹 셋을 유지~~
5. ~~부족한 자원 증설~~
6. ~~일부 프로세스 중단~~

A 4-3-8 프로세스의 개요

프로세스의 정의

1. 프로세스가 할당되는 실체로, 디스패치가 가능한 단위
2. 비동기적 행위를 일으키는 주체
3. PCB를 가지며 PCB에는 프로세스의 현재 상태, 고유 식별자를 가지고 있다.
4. ~~실기억장치에 저장된 프로그램~~
5. ~~프로시저가 활동중인 것~~
6. ~~운영체제가 관리하는 실행 단위~~

PCB

1. 프로세스 식별자, 프로세스 상태 등의 정보로 구성

프로세스 상태 전의

Submit (제출) > Hold (접수) > Ready (준비) > Running (실행) > Wait (대기, 보류), Block (블록) > Exit (종료)

프로세스 상태 전의 관련 용어

1. 디스패치 (Dispatch) : 프로세스가 준비 상태에서 프로세서가 배당되어 실행 상태로 변화하는 것
2. ~~Wake Up, Spooling, 교통량 제어기~~

스레드 : 프로세스의 작업 단위

1. 사용자 수준 :
    A. 커널 모드로의 전환이 없이 오버헤드가 줄어든다.
    B. 사용자가 만든 라이브러리를 사용하여 스레드를 운용
2. 커널 수준 :
    A. 운영체제에 의해 스레드를 운용
3. 스레드 사용의 장점
    A. 하드웨어, 운영체제의 성능과 응용 프로그램의 처리율을 향상
    B. 버헤드가 줄어든다.

A 4-3-10 주요 스케줄링 알고리즘

SJF : 실행 시간이 가장 짧은 프로세스에게 먼저 CPU을 할당

HRN : 대기 시간과 서비스(실행) 시간을 이용하는 기법

1. 우선순위가 높은 것부터 실행
2. 우선순위 = (대기시간 + 서비스 시간) / 서비스 시간
3. 대기시간이 길면 우선순위가 높다
4. 서비스 시간이 짧으면 우선순위가 높다
5. SJF 기법을 보안하기 위한 방식
6. 긴 작업과 짧은 작업 간의 지나친 불평등을 해소

B 4-3-11 환경 변수

UNIX / LINUX의 주요 환경 변수

1. 출력 명령어 : env, printenv, setenv, set
2. 환경변수 : export
    A. export가 매개변수 없이 쓰일 경우, 현재 설정된 환경변수들이 출력
    B. 사용자가 생성하는 변수는 export 명령어 표시하지 않는 한 현재 쉘에 국한
    C. 변수를 export 시키면 전역변수처럼 되어 끝까지 기억

B 4-3-12 운영체제 기본 명령어

UNIX / LINUX 기본 명령어

1. Fork : 새로운 프로세스 생성
2. Uname : 버전을 확인 / 시스템 정보를 표시
3. Cat, Chdir, Chmod, Chown, Cp, Exec, Find, Fsck, Getpid, Getppid, Is, Mount/unmounts, Rm, wait

A 4-3-13 인터넷

IP 개요

1. IPv6의 패킷 헤더는 40Byle(옥텟 octet)의 고정된 길이를 갖는다
2. IPv6는 주소 자동설정 기능을 통해 손쉽게 이용자의 단말을 네트워크에 접속
3. IPv4는 호스트 주소를 자동으로 설정하며 유니케이스를 지원
4. IPv4는 클래스별로 네트워크와 호스트 주소의 길이가 다르다

IP 주소

1. A Class : 0~127 / 국가나 대형 통신망에 사용
2. B Class : 128~191 / 중대형 통신망에 사용
3. C Class : 192~233 / 소규모 통신망에 사용
4. D Class : 224~239 / 멀티캐스트 용으로 사용
5. E Class : 실험적 주소

IPv6의 개요

1. 128비트의 주소 공간을 제공
2. 인증 및 보안 기능을 포함
3. 패깃 크기 제한이 없다
4. 확장 헤더를 통해 네트워크 기능 확장이 용이

IPv6의 구성

1. Unicast (유니케이스)
2. Multicast (멀티캐이스)
3. Anycast (애니캐이스) : 하나의 호스트에서 그룹 내의 가장 가까운 곳에 있는 수신자에게 전달하는 방식

IPv6의 특성

1. 16비트씩 8부분의 16진수로 표시 / 2^{128}개의 주소를 표현할 수 있다.
2. 등급별, 서비스별로 패킷을 구분할 수 있어 품질보장이 용이
3. 확장기능을 통해 보안기능을 제공

A 4-3-14 OSI 참조 모델

물리 계층 : 비트

데이터 링크 계층 : 프레임

1. 동기화, 오류제어, 흐름제어 등의 전송 에러를 제어하는 계층
2. 링크의 설정과 유지 및 종료를 담당
3. 노드간의 오류 제어와 흐름 제어 기능을 수행
4. HDLC, PPP, LLC, ~~LAPB, MAC, PAPD~~ 등의 표준

네트워크 계층 : 패킷

1. 패킷을 발신지로부터 최종 목적지까지 전달하는 책임을 진다
2. 패킷에 발신지와 목적지의 논리 주소를 추가
3. 라우터 또는 교환기는 패킷 전달을 위해 경로를 지정하거나 교환 기능을 제공

전송 계층 : 세그먼트

1. 단말기 사이에 오류 수정과 흐름 제어를 수행하여 신뢰성 있고 명확한 데이터를 전달

세션 계층 : 메시지

1. 두 응용 프로세스 간의 통신에 대한 제어 구조를 제공
2. 연결의 생성, 관리, 종료를 위해 토큰을 사용

B 4-3-15 네트워크 관련 장비

브리지

1. LAN과LAN을 연결
2. LAN 안에서의 컴퓨터 그룹을 연결
3. 데이터 링크 계층 중 MAC계층

스위치

1. LAN과 LAN을 연결하여 훨씬 더 큰 LAN을 만드는 장치
2. OSI 7계층의 데이터 링크 계층에서 사용

라우터

1. 두 개의 서로 다른 형태의 네트워크를 상호 접속하는 3계층 장비
2. LAN과 LAN의 연결 기능에 데이터 전송의 최적 경로를 선택할 수 있는 기능이 추가
3. 서로 다른 LAN이나 LAN과 WAN의 연결도 수행
4. OSI 7계층의 네트워크 계층에서 동작

A 4-3-17 TCP / IP

개요

1. TCP
    A. 신뢰성 있는 연결 지향형 전달 서비스
    B. 스트림 전송 기능을 제공
    C. 순서 제어, 오류제어, 흐름 제어 기능을 제공
    D. 기본 헤더 크기 20~60byte이지만 최대 100byte까지 확장 가능
2. IP
    A. 패킷을 분할, 병합하는 기능을 수행
    B. 비연결형 서비스 제공
    C. Best Effort 원칙에 따른 전송 기능을 제공
    D. IP 헤더에서 제공하는Checksum(체크섬)은 Header Checksum, ~~Version, Header Length, Total~~ 등 이다

TCP헤더

1. 순서번호 : 전달하는 바이트마다 번호가 부여
2. 수신번호확인 : 상대편 호스트에서 받으려는 바이트의 번호를 정의
3. 체크섬 : 데이터를 포함한 세그먼트의 오류를 검사
    A. Header Length : IP 프로토콜의 헤더 길이를 32비트 워드 단위로 표시
    B. Time To Live : 송신 호스트가 패킷을 전송하기 전 네트워크에서 생존할 수 있는 시간을 지정
    C. Version Number : IP 프로토콜의 버전번호를 나타낸다
    D. Packet Length : IP 패킷 전체 길이를 바이트 단위로 길이를 표시 (최대크기 $2^{16} - 1$ 비트)
4. 윈도우 크기는 송수신 측의 버퍼 크기로 최대크기는 16비트로 2^16 = 65535byte = 64KB.

구조

1. 응용 계층 : HTTP, DNS
    A. 응용 계층 : 사용자가 OSI환경에 접근할 수 있도록 서비스 제공
    B. 표현 계층 : 통신에 적당한 형태로 변환

C. 세션 계층 : 대화 제어를 담당

2. 전송 계층 (트렌스포트 계층) : TCP , ~~UDP , RTCP~~

A. 전송 계층 : 오류검출과 복구, 흐름 제어를 수행하는 계층

3. 인터넷 계층 : IP , ARP(논리>물리) , RARP(물리>논리)

A. 네트워크 계층 : 네트워크 연결을 관리 및 데이터 교환 및 중계

4. 네트워크 액세스 계층

A. 데이터 링크 계층 : 신뢰성있고 효율적인 정보 전송 : HDLC, PPP, LLC

B. 물리 계층 : 실제 접속과 절단 등 기계적, 전기적 기능적 절차 특성에 대한 규칙

응용 계층의 주요 프로토콜

1. MQTT : IBM이 주도하여 개발

A. 발행-구독 기반의 메시징 프로토콜

B. 경량 메시지 전송 프로토콜

C. IoT 환경에서 자주 사용

전송 계층의 주요 프로토콜

1. TCP

A. 흐름 제어 기능을 수행

B. 전이 중(Full Duplex) 방식의 양방향 가상회선을 제공

C. 전송 데이터와 응답 데이터를 함께 전송

2. UDP

A. 흐름제어, 순서제어가 없어 전송속도가 빠르다

B. 비연결형 서비스 제공

C. 단순한 헤더 구조로 오버헤드가 적다

D. TCP와 같이 트랜스포트 계층에 존재

~~3. RTCP~~

인터넷 계층의 주요 프로토콜

1. ICMP : 전송 오류가 발생하는 경우에 대비해 오류 전송을 전송하는 목적으로 사용

2. ARP : 논리 주소 > 물리 주소 변환

3. RARP : 물리 주소 > 논리 주소 변환

~~4. IP~~

~~5. IGMP~~

네트워크 액세스 계층의 주요 프로토콜

1. Ethernet(IEEE 802.3) : CSMA/CD 방식의 LAN에서 사용되는 전송 매체 접속 제어(MAC) 방식

~~2. IEEE 802~~

~~3. HDLC~~

~~4. X.25~~

~~5. RS-232C~~

5과목 정보시스템 구축 관리

5-1 소프트웨어 개발 방법론 활용

B 5-1-1 소프트웨어 개발 방법론

구조적 방법론 : 검토>계획>요구사항>설계>구현>시험>유지보수

1. 정형화된 분석 절차에 따라 사용자 요구사항을 파악, 문서화 하는 체계적 분석방법
2. 자료흐름도 DFD, 자료사전 DD, 소단위명세서의 특징을 갖는다
3. 분할과 정복

정보공학 방법론 : 계획>분석>설계>구축

1. 데이터 설계 DRD 사용

컴포넌트 기반 CBD : 준비>분석>설계>구현>테스트>전개>인도

1. 개발 기간 단축으로 인한 생산성 향상
2. 새로운 기능 추가가 쉬운 확장성
3. 소프트웨어 재사용이 가능
4. 생산성과 품질을 높임
5. 유지보수 비용을 최소화
6. 컴포넌트 제작 기법을 통해 재사용성 향상
7. 독립적인 컴포넌트 단위의 관리로 복합성을 최소화

컴포넌트 기반 CBD 개발 공정 :

1. 분석 단계 : 요구사항 정의서, ~~유스케이스 명세서, 요구사항 추적표~~
2. ~~설계 : {클래스, UI, 아키텍처, 인터페이스, 컴포넌트, 데이터베이스} 설계서~~
3. ~~구현 : 프로그램 코드, 단위시험 결과서, 데이터베이스 테이블~~
4. ~~테스트 : {통합시험, 시스템 설치, 시스템시험, 인수시험} 결과서, {사용자, 운영자} 지침서~~

A 5-1-2 S/W 공학의 발전적 추세

소프트웨어 재사용의 개요

1. 장점
    A. 위험부담 감소
    B. 비용절감
    C. 시스템 명세의 오류 억제
    D. 품질 향상
    E. 생산성 향상
    F. 시스템 명세, 설계 코드 등의 문서를 공유

소프트웨어 재사용 방법

1. 합성 중심(블록구성) : 전자 칩과 같은 소프트웨어 부품. 즉 블록(모듈을 만들어서 끼워 맞추는 방법)
2. ~~생성 중심(패턴구성) : 추상화 형태로 쓰여진 명세를 구체화하여 프로그램을 만드는 방법~~

소프트웨어 재공학의 개요

1. Migration(이식) : 기존 소프트웨어를 다른 환경에서 사용할 수 있도록 변환
2. 분석(Analysis)
3. 재구성(Restructuring)
4. 역공학(Reverse Engineering)

CASE의 개요

1. 소프트웨어 모듈의 재사용성이 향상
2. 자동화된 기법을 통해 소프트웨어 품질이 향상
3. 소프트웨어 유지보수를 간편하게 수행

CASE의 원천 기술

1. 구조적 기법
2. 프로토타이핑 기술
3. 정보 저장소 기술

~~4. 자동 프로그래밍~~

~~5. 분산처리~~

CASE의 주요 기능

1. 모델들 사이의 모순검사
2. 모델의 오류검증
3. 자료 흐름도 작성
4. 소프트웨어 생명 주기 전 단계의 연결
5. 다양한 소프트웨어 개발 모형 지원
6. 그래픽 지원

A 5-1-5 비용 산정 기법 – 상향식

LOC 기법 : 원시 코드 라인 수

원시 코드 라인수의 비관치, 낙관치, 기대치를 측정하여 예측치를 구하고 이를 이용하여 비용을 산정

1. 예측치 = (a + 4m + b) / 6 || a-낙관치, b-비관치, m-기대치(중간치)
2. 산정 공식
   A. 노력(인월) = 개발시간 * 투입인원 = LOC / 1인당 월평균 생산 코드 라인 수
   B. 개발 비용 = 노력(인월) * 단위 비용(1인당 월평균 인건비)
   C. 개발 기간 = 노력(인월) / 투입 인원
   D. 생산성 = LOC / 노력(인월)

A 5-1-6 수학적 산정 기법 (소프트웨어 비용 추정모형)

COCOMO 모형 비용 산정

1. 보헴이 제안한 원시 프로그램의 규모에 의한 비용 예측 모형
2. 같은 규모의 소프트웨어라도 그 유형에 따라 비용이 다르게 산정
3. 프로젝트를 완성하는데 필요한 man-month로 산정 결과를 나타낼 수 있다.

COCOMO의 소프트웨어 개발 유형 : 중소 규모 소프트웨어 프로젝트 비용 추정에 적합

1. Organic Mode (조직형) : 5만 라인 이하
2. Semi-detached Mode (반분리형) : 30만 라인 이하
3. Embedded Mode (내장형) : 30만 라인 이상

Putnam 모형

1. Rayleigh-Norden 곡선의 노력 분포도를 이용한 프로젝트 비용 산정 기법

Function-point(FP) 기능 점수 모형

1. 기능별 가중치
    A. 명령어(사용자 질의수)
    B. 데이터 파일
    C. 출력 보고서
    ~~D. 자료 입력(입력 양식)~~
    ~~E. 필요한 외부 루틴과의 인터페이스~~
2. 자동화 추정 도구
    A. SLIM : Putnam 기법 모형을 기초로 개발된 자동화 추정 도구
    B. ESTIMACS

B 5-1-7 프로젝트 일정 계획

PERT

1. 작업들 간의 상호 관련성, 경정 경로, 경계 시간, 자원 할당 등을 제시

CPM : 최장 경로

1. 프로젝트 내에서 각 작업이 수행되는 시간과 각 작업 사이의 관계를 파악
2. 효과적인 프로젝트 통제
3. 경영층 과학적인 의사 결정을 지원

간트 차트

1. 프로젝트를 이루는 소작업별로 언제 시작되고 언제 끝나야 하는지 한 눈에 볼 수 있다.
2. 자원 배치 계획에 유요하게 사용
3. CPM 네트워크로부터 만드는 것이 가능
4. 수평 막대의 길이는 각 작업의 기간

B 5-1-8 소프트웨어 개발 방법론 결정

프로젝트 관리

1. 주어진 기간 내에 최소의 비용으로 사용자를 만족시키는 시스템을 개발

A 5-1-9 소프트웨어 개발 표준

ISO/IEC 12207

1. 기본 생명 주기 프로세스 : 획득, ~~공급~~, 개발, ~~운영~~, 유지보수
2. 지원 생명 주기 프로세스 : 품질 보증, 검증, 확인, 활동 검토, 감사, 문서화, 형상관리, 문제 해결
3. 조직 생명 주기 프로세스 : 관리, 기반 구조, 훈련, 개선

CMM : 초기>관리>정의>정량적관리>최적화

SPICE : 소프트웨어 품질 및 생산성 향상을 위해 소프트웨어 프로세스를 평가 및 개선하는 국제 표준

1. ~~수준 0 – 불완전~~
2. ~~수준 1 – 수행~~
3. ~~수준 2 – 관리~~
4. 수준 3 – 확립
5. 수준 4 – 예측
6. 수준 5 – 최적화

B 5-1-10 소프트웨어 개발 방법론 테일러링

개요

1. 프로젝트에 최적화된 개발 방법론을 적용하기 위해 절차, 산출물 등을 적절히 변경
2. 관리적 측면 - 최단기간에 안정적인 프로젝트 진행을 위한 사전 위험을 식별하고 제거하는 것
3. 기술적 측면 - 최적화된 기술 요소를 도입하여 프로젝트 특성에 맞는 최적의 기법과 도구를 사용

고려사항

1. 내부적 기준
    A. 목표 환경 : 납기/비용, 기술환경, 구성원 능력, 고객요구 사항
    B. 요구사항
    C. 프로젝트 규모 : 비용, 인력, 기간 등
    D. 보유 기술 : 구성원의 능력, 프로세스, 개발 방법론 등
2. 외부적 기준
    A. 법적 제약사항
    B. 표준 품질 기준

A 5-1-11 소프트웨어 개발 프레임워크

설명

1. 반제품 상태의 제품을 토대로 도메인 별로 필요한 서비스 컴포넌트를 사용
2. 재사용성 확대와 성능을 보장 받을 수 있게 하는 개발 소프트웨어이다.
3. 생산성 향상과 유지보수성 향상 등의 장점이 있다.

특성

1. 모듈화 : 품질 향상, 유지보수 용이
2. 재사용성 : 예산 절감, 생산성 향상, 품질 보증
3. 확장성
4. 제어의 역흐름 : 외부 라이브러리의 코드를 호출해 이용, 제어를 프레임워크에 넘김으로써 생산성을 향상

5-2 IT프로젝트 정보시스템 구축 관리

A 5-2-1 네트워크 관련 신기술

소프트웨어 정의 기술

1. 소프트웨어 정의 네트워킹 : SDN
    A. 네트워크를 제어부, 데이터 전달부로 분리
    B. 효율적으로 네트워크를 제어, 관리
    C. 안전성, 속도, 보안 등 소프트웨어로 제어 관리하기 위해 개발
    D. 기존 네트워크에는 영향을 주지 않음
    E. 특정 서비스의 전송 경로 수정을 통하여 인터넷상에서 발생하는 문제를 처리
2. 소프트웨어 정의 데이터 센터 : SDDC
    A. 컴퓨팅, 네트워킹, 스토리지, 관리 등을 모두 소프트웨어로 정의
    B. 인력 개입 없이 소프트웨어 조작만으로 자동 제어 관리
    C. 데이터 센터 내 모든 자원을 가상화하여 서비스
3. Software Defined Storage : SDS : 소프트웨어 정의 스토리지
    A. 가상화를 적용하여 필요한 공간만큼 나눠 사용할 수 있도록 하며 서버 가상화와 유사
    B. 데이터 스토리지 체계
    C. 일정 조직 내 여러 스토리지를 하나처럼 관리하고 운용하는 컴퓨터 이용 환경
    D. 스토리지 자원을 효율적으로 나누어 쓰는 방법

IoT 관련 용어

1. Mesh Network (매시 네트워크) :
    A. 기존 무선 랜의 한계 극복을 위해 등장
    B. 대규모 디바이스의 네트워크 생성에 최적화
    C. 직접 접속되는 그물 모양의 네트워크
    D. 대용량을 빠르고 안전하게 전달

2. PICONET (피코넷) :

    A. 여러 개의 독립된 통신장치가 UWB 기술 또는 블루투스 기술을 사용하여 통신망을 형성

클라우드 컴퓨팅

1. Pass-TA, PaaS-TA (파스-타) :

    A. 국내 IT 서비스 경쟁력 강화를 목표로 개발

    B. 인프라 제어 및 관리 환경, 실행 환경, 개발 환경, 서비스 환경, 운영 환경으로 구성

    C. 개방형 클라우드 컴퓨팅 플랫폼

2. 클라우드 기반 HSM :

    A. 클라우드(데이터센터) 기반 암호화 키 생성, 처리, 저장 등을 하는 보안 기기

    B. 국내에서는 공인인증제의 폐지와 전자서명법 개정을 추진하면서 클라우드 HSM 용어가 자주 등장

    C. 클라우드에 인증서를 저장하므로 기존 HSM 기기나 휴대폰에 인증서를 저장해 다닐 필요가 없다

    D. 하드웨어적으로 구현되므로 소프트웨어식 암호 기술에 내재된 보안 취약점을 해결

기타 용어

1. WDM : Wavelength Division Multiplexing : 파장 분할 다중화

    A. 광섬유를 이용한 통신 기술

    B. 파장이 서로 다른 복수의 광신호를 동시에 이용하는 것 (광섬유를 다중화 하는 방식)

    C. 빛의 파장 축과 파장이 다른 광선은 서로 간섭을 일으키지 않는 성질

2. SSO :

    A. 시스템이 개수상관 없이 하나의 시스템에서 인증에 성공하면 다른 시스템에 대한 접근 권한도 얻음

3. 스마트 그리드

    A. 전기 및 정보통신기술을 활용하여 전력망을 지능화, 고도화 함

    B. 고품질의 전력 서비스를 제공하고 에너지 이용효율을 극대화 하는 전력망

4. Zing

    A. 기기를 키오스크에 갖다 대면 원하는 데이터를 바로 가져올 수 있는 기술

    B. 10㎝ 이내 근접 거리에서 기가급 속도로 데이터 전송이 가능한 초고속 근접무선통신(NFC) 기술

A 5-2-2 네트워크 구축

버스형

1. 한 개의 통신 회선에 여러 대의 단말장치가 연결되어 있는 형태

VLAN

1. 물리적 배치와 상관없이 논리적으로 LAN을 구성
2. 접속된 장비들의 성능 향상 및 보안성 증대

LAN의 표준안

1. IEEE 802.3 : CSMA/CD
2. IEEE 802.11a : 5GHz 대역 전파 사용, OFDM 기술 사용
3. IEEE 802.11b : 전송 속도 최고 11Mbps
4. IEEE 802.11e : QoS 강화를 위해 MAC(매체 접근 제어) 지원 기능
5. IEEE 802.11g : 2.4GHz 대역 전파 사용
6. IEEE 802.11i : WPA/WPA2(Wi-Fi 인증 및 암호화) 사용
7. IEEE 802.11n : 2.4GHz 대역과 5GHz 대역 사용

CSMA/CA : 무선랜 사용

1. 무선 랜에서 데이터 전송 시 매체가 비어있음을 확인한 뒤 충돌을 회피하기 위해 임의 시간을 기다린 후 데이터를 전송하는 방법
2. 네트워크에 데이터의 전송이 없는 경우라도 동시 전송에 의한 충돌에 대비하여 확인 신호를 전송

A 5-2-4 경로 제어 / 트래픽 제어

경로 제어 프로토콜

1. IGP
    A. RIP
        i. 거리 벡터 라우팅 프로토콜
        ii. 소규모 네트워크 환경에 적합
        iii. 최단 경로 탐색에는 Bellman-ford 알고리즘 사용
        iv. 최대 홉 카운트 15로 제한
        v. 경로 선택 메트릭은 홉 카운트
        vi. 각 라우터는 이웃 라우터들로부터 수신한 정보를 이용하여 라우팅 표를 갱신
    B. OSPF
        i. 네트워크 변화에 신속하게 대처
        ii. 멀티캐스팅을 지원
        iii. 최단 경로 탐색에 Dijkstra 알고리즘을 사용
2. EGP
3. BGP

흐름 제어

1. Stop and Wait (정지 - 대기) :
    A. 프레임이 손실되었을 때, 손실된 프레임 1개를 전송하고 수신자의 응답을 기다리는 방식
    B. 한 번에 프레임 1개만 전송할 수 있는 기법

~~2. Sliding Window (슬라이딩 윈도우) :~~

A 5-2-5 SW 관련 신기술

SW 관련 용어

1. Mashup(매쉬업) : 정보 및 서비스를 이용하여 새로운 소프트웨어나 서비스, 데이터베이스 등을 만듦
2. SOA(서비스 지향 아키텍처)
3. 디지털 트윈(Digital Twin) : 물리적인 자산 대신 소프트웨어로 가상화
4. 텐서플로(TensorFlow) :
    A. 구글의 구글 브레인 팀이 제작
    B. 기계 학습을 위한 오픈 소스 소프트웨어 라이브러리
5. Docker(도커) :
    A. 컨테이너 응용 프로그램의 배포를 자동화하는 오픈소스 엔진
    B. 소프트웨어 컨테이너 안에 응용 프로그램들을 배치시키는 일을 자동화
6. Scrapy(스크래피) : Python 기반의 웹 크롤링 프레임워크

A 5-2-6 보안 관련 신기술

보안 관련 용어

1. BaaS (서비스형 블록체인)
    A. 블록체인 개발 환경을 클라우드로 서비스 하는 개념
    B. 블록체인 네트워크에 노드의 추가 및 제거가 용이
    C. 블록체인의 기본 인프라를 추상화하여 블록체인 응용 프로그램을 만듦
2. OWASP
    A. 오픈소스 웹 애플리케이션 보안 프로젝트
    B. 웹을 통한 정보 유출, 악성 파일 및 스크립트, 보안 취약점 등을 연구
3. TCP Wrapper(TCP 래퍼)
    A. 외부 컴퓨터가 접속되면 접속 인가 여부를 점검해서 인가된 경우 접속이 허용 / 반대의 경우 거부
4. Honeypot 허니팟 :
    A. 1990년대 David Clock이 처음 제안
    B. 비정상적인 접근의 탐지를 위해 의도적으로 설치해 둔 시스템
    C. 침입자를 속여 실제 공격당하는 것처럼 보여줌
    D. 크래커를 추적 및 공격기법의 정보를 수집하는 역할
    E. 쉽게 공격자에게 노출
    F. 쉽게 공격이 가능한 것처럼 취약해 보여야 함
5. DPI
    A. 프로토콜과 패킷 내부의 콘텐츠를 파악하여 침입 시도, 해킹 등을 탐지
    B. 트래픽을 조정하기 위한 패킷 분석 기술
    C. 침입 시도, 해킹 등을 탐지하고 트래픽을 조정

B 5-2-9 HW 관련 신기술

고가용성 솔루션(HACMP)

1. 각 시스템 간에 공유 디스크를 중심으로 클러스터링으로 엮어 다수의 시스템을 동시에 연결
2. 조직, 기업의 기간 업무 서버 안정성을 높이기 위해 사용
3. 2개의 서버를 연결하는 것으로 2개의 시스템이 각각 업무를 수행하도록 구현하는 방식

N-Screen(앤 스크린)

1. PC, TV, 휴대폰에서 원하는 콘텐츠를 끊김없이 자유롭게 이용할 수 있는 서비스

B 5-2-10 Secure OS

개요

1. 운영체제의 커널에 보안 기능을 추가한 것
2. 운영체제의 보안상 결함으로 인하여 발생 가능한 각종 해킹으로부터 시스템을 보호하기 위하여 사용

보안 기능

1. 식별 및 인증
2. 임의적 접근통제
3. 강제적 접근통제
4. ~~객체 재사용 보호~~
5. ~~완전한 조정~~
6. ~~신뢰 경로~~
7. ~~감사 및 감사기록 축소~~

Secure 코딩에서 입력 데이터의 보안 약점

1. SQL 삽입 : 사용자의 입력 값 등 외부 입력 값이 SQL 쿼리에 삽입되어 공격
2. 크로스사이트 스크립트 : 검증되지 않은 외부 입력 값에 의해 브라우저에서 악의적인 코드가 실행
3. 운영체제 명령어 삽입 : 운영체제 명령어 파라미터 입력 값이 적절한 사전검증을 거치지 않고 사용되어 공격자가 운영체제 명령어를 조작
4. 자원 삽입 : 자원을 조작 할 수 있는 문자열을 삽입하여 시스템이 보호하는 자원에 임의로 접근 할 수

A 5-2-11 DB 관련 신기술

Hadoop (하둡)

1. 오픈 소스를 기반으로 한 분산 컴퓨팅 플랫폼
2. 일반 PC급 컴퓨터들로 가상화된 대형 스토리지를 형성
3. 다양한 소스를 통해 생성된 빅데이터를 효율적으로 저장하고 처리
4. 하둡과 관계형 데이터베이스 간에 데이터를 전송할 때 Sqoop(스쿱) 도구 사용

MapReduce (맵리듀스)

1. 대용량 데이터를 분산 처리하기 위한 목적으로 개발된 프로그래밍 모델
2. 구글에 의해 고안된 기술로서 대표적인 대용량 데이터를 처리를 위한 병렬 처리 기법을 제공
3. 임의의 순서로 정렬된 데이터를 분산 처리하고 이를 다시 합치는 과정을 거친다

Data Mining (데이터 마이닝)

1. 대량의 데이터를 분석하여 데이터 속에 내재되어 있는 변수 사이의 상호관계를 규명하여 패턴화

OLAP (Online Analytical Processing) 연산

Roll-up, Dicing, Drill-down, ~~Drill-through, Drill-across, Pivoting, Slicing~~

Honeypot 허니팟

1. 1990년대 David Clock이 제안했다
2. 의도적으로 설치해 둔 시스템
3. 침입자를 속여 실제 공격당하는 것처럼 보여줌
4. 크래커를 추적 및 공격기법의 정보를 수집하는 역할
5. 취약한 모습을 보여야 함

Windows 파일 시스템

1. FAT
2. NTFS
    A. 대용량 볼륨에 효율적
    B. 자동 압축 및 안정성

C. 저용량 볼륨에서의 속도 저하

D. 보안이 FAT보다 뛰어남

A 5-2-12 회복 / 병행제어

Recovery (회복)

1. 트랜잭션들을 수행하는 도중 장애로 인해 손상된 데이터베이스를 손상되기 이전의 정상적인 상태로 복구

회복 기법

1. 즉각 갱신 기법 : 데이터베이스 로그(log)를 필요로 하는 회복 기법
2. ~~연기 갱신 기법~~
3. ~~그림자 페이지 대체 기법~~
4. ~~검사점 기법~~

병행제어 목적

1. 시스템 활용도 최대화
2. 사용자에 대한 응답시간 최소화
3. 데이터베이스 일관성 유지

병행제어 기법의 종류

1. 로킹 :
2. 타임 스탬프 순서 : 동시성 제어를 위한 직렬화 기법으로 트랜잭션 간의 처리 순서를 미리 정하는 방법
3. ~~최적 병행수행~~
4. ~~다중 버전 기법~~

로킹 단위

1. 로킹의 대상이 되는 객체의 크기를 로킹 단위라고 한다
2. 데이터베이스, 파일, 레코드 등은 로킹 단위가 될 수 있다
3. 로킹단위↑ : 로크 수↓, 병행성↓, 오버헤드↓, 공유도↓, 제어기법 간단하여 관리 수월
    A. 병행성 수준이 낮아짐, 공유도 감소
4. 로킹단위↓ : 로크 수↑, 병행성↑, 오버헤드↑, 공유도↑, 제어기법 까다로워 관리 복잡
    A. 병행성 수준 높아짐, 공유도 증가

B 5-2-13 교착상태

발생의 필요 충분 조건

1. Mutual exclusion 상호 배제
2. Hold and wait 점유와 대기
3. Circular wait 환형 대기
4. Non-preemption 비선점

해결 방안

1. Avoidance 회피 기법 : 은행 알고리즘
2. ~~Prevention 예방 기법 : 교착 상태의 원인이 되는 조건 중 하나를 제거~~
3. ~~Detection 발견 기법 : 자원 할당 그래프~~
4. ~~Recovery 회복(복구) 기법 : 자원 선점/프로세스 종료~~

5-3 소프트웨어 개발 보안 구축

A 5-3-1 Secure SDLC

개요

1. Seven Touchpoints : SW 보안의 모범사례를 SDLC에 통합한 방법론
2. ~~CLASP~~
3. ~~SDL~~

정보보안 3요소

1. 무결성(수정) : 인가된 사용자만 수정 가능 / 전송중인 정보는 수정되지 않는다
2. 기밀성(열람) :
3. 가용성(사용) : 인가된 사용자는 가지고 있는 권한 범위 내에서 언제든 자원 접근이 가능

B 5-3-2 세션 통제

세션 하이재킹 탐지 방법

1. 비동기화 상태 탐지
2. ACK STORM 탐지
3. 패킷의 유실 및 재전송 증가 탐지
4. 예상치 못한 접속의 리셋 탐지

A 5-3-3 입력 데이터 검증 및 표현

보안 약점 종류

1. SQL Injection (삽입)
    A. 임의로 작성한 SQL 구문을 애플리케이션에 삽입하는 공격 방식
    B. 취약점이 발생하는 곳은 주로 웹 애플리케이션과 데이터베이스가 연동되는 부분
    C. 로그인과 같이 웹에서 사용자의 입력 값을 받아 데이터베이스 SQL문으로 데이터를 요청하는 경우
    D. 사용자의 입력 값 등 외부 입력 값이 SQL 쿼리에 삽입되어 공격
2. 경로 조작 및 자원 삽입
3. XXS (크로스사이트 스크립팅)
    A. 웹페이지에 악의적인 스크립트를 포함시켜 사용자 측에서 실행되게 유도함
    B. 정보 유출 등의 공격을 유발할 수 있는 취약점
    C. 검증되지 않은 외부 입력 값에 의해 브라우저에서 악의적인 코드가 실행
4. 운영체제 명령어 삽입
    A. 파라미터 입력값이 적절한 사전검증을 거치지 않고 사용되어 공격자가 운영체제 명령어를 조작
5. 메모리 버퍼 오버플로
    A. 메모리를 다루는 데 오류가 발생하여 잘못된 동작을 하는 프로그램 취약점

~~6. 위험한 형식 파일 업로드~~

~~7. 신뢰되지 않는 URL 주소로 자동접속 연결~~

리눅스

1. wtmp
    A. 사용자의 성공한 로그인/로그아웃 정보기록
    B. 시스템의 종료/시작 시간 기록

B 5-3-4 보안 기능

보안 기능의 보안 약점

1. 하드코드 된 비밀번호
2. ~~적절한 인증 없이 중요기능 허용~~
3. ~~부적절한 인가~~
4. ~~중요한 자원에 대한 잘못된 권한 설정~~
5. ~~취약한 암호화 알고리즘 사용~~
6. ~~중요정보 평문 저장 및 전송~~

B 5-3-7 코드 오류

스택 가드

메모리상에서 프로그램의 복귀 주소와 변수 사이에 특정 값을 저장해 두었다가

그 값이 변경되었을 경우, 오버플로우 상태로 가정하여 프로그램 실행을 중단하는 기술

B 5-3-8 캡슐화

접근 제어자 종류

1. Public
2. Protected
3. Default
4. private

A 5-3-10 암호화 알고리즘

개요

1. 암호화 방식
    A. 단방향(비대칭)
        i. HASH 해시
    B. 양방향(대칭)
        i. 공개키
        ii. 개인키
            1. Stream 방식
            2. Block 방식

개인키 암호화 기법 (대칭)

1. 종류
    A. 블록 암호화 방식 : AES, IDEA, DES, SEED, ~~ARIA~~
    B. 스트림 암호화 방식
        i. RC4, LFSR
        ii. 비트/바이트/단어들을 순차적으로 암호화
        iii. 대칭키 암호화 방식
2. 장점 :
    A. 실행 속도가 빠르다
    B. 다양한 암호의 핵심 함수로 사용
3. ~~단점 :~~

공개키 암호화 기법 (대칭)

1. 비대칭 암호 기법
2. 대표적인 기법 = RSA
3. 키 분배가 용이하고, 관리해야 할 키 개수가 적다

4. 복호화키 = 비공개
5. 송수신자는 수신자의 공개키로 문서를 암호화

양방향 (대칭) 알고리즘 종류

1. RSA : 큰 숫자를 소인수 분해하기 어렵다는 기반하에 1978년 MIT에 의해 제안된 공개키
2. AES : 암호화 키와 복호화 키가 동일하다
3. DES : 64비트의 암호화 알고리즘
4. IDEA

해시 (Hash)

1. 임의의 길이의 입력 데이터를 받아 고정된 길이의 해쉬 값으로 변환
2. 대표적인 기법 = HAVAL, SHA-1, MD4, MD5 등
3. 해쉬 함수는 일방향 함수 (One-way function)

Salt (솔트)

1. 시스템에 저장되는 패스워드들은 Hash 또는 암호화 알고리즘의 결과 값으로 저장
2. 암호 공격을 막기 위해 똑 같은 패스워드들이 다른 암호 값으로 저장되도록 추가되는 값

대칭 암호

1. 실행 속도가 빠르기 때문에 다양한 암호의 핵심 함수로 사용

비대칭 암호

1. 자신만이 보관하는 비밀키를 이용하여 인증, 전자서명 등에 적용

5-4 시스템 보안 구축

A 5-4-1 서비스 공격 유형

Smurfing (스머핑)

1. 특정 사이트에 집중적으로 데이터를 보내 네트워크 또는 시스템의 상태를 불능으로 만드는 공격 방법

DDoS 공격

1. Tribe Flood Network : TFN
2. ~~Trin00~~
3. ~~TFN2K~~
4. ~~Stacheldraht~~

네트워크 침해 공격 관련 용어

1. Phishing , Evil Twin Attack
    A. 소셜 네트워크에서 악의적인 사용자가 지인 또는 특정 유명인으로 가장하여 활동하는 공격 기법
2. Ping Flood
    A. 특정 사이트에 매우 많은 ICMP Echo를 보내면, 이에 대한 응답을 하기 위해 시스템 자원을 모두 사용해버려 시스템이 정상적으로 동작하지 못하도록 하는 공격방법
3. Ping of Death :
    A. 허용범위 이상의 ICMP 패킷을 전송하여 대상 시스템의 네트워크를 마비
4. 스위치 재밍

블루투스 관련 공격

1. 블루프린팅 : 블루투스 공격 장치의 검색 활동을 의미
2. ~~블루버그~~
3. ~~블루스나프~~
4. ~~블루재킹~~

정보 보안 침해 공격 관련 용어

1. Worm (웜)

A. 컴퓨터의 취약점을 이용하여 스스로 전파하거나 메일로 전파되며 스스로 증식

2. Key Logger Attack (키로거 공격)

A. 컴퓨터 사용자의 키보드 움직임을 탐지해 ID, 패스워드 등 개인의 중요한 정보를 몰래 빼가는 해킹

3. Ransomware (랜섬웨어)

A. 인터넷 사용자의 컴퓨터에 침입해 내부 문서 파일 등을 암호화해 사용자가 열지 못하게 하는 공격

B. 암호 해독용 프로그램의 전달을 조건으로 사용자에게 돈을 요구

4. 백도어 : ~~설계자가 서비스 기술자의 편의를 위해 보안을 제거하여 만들어놓은 비밀 통로~~

A. 무결성 검사

B. 로그 분석

C. SetUID 파일 검사

D. 열린 포트 확인

~~E. 바이러스 및 백도어 탐지 툴 이용~~

취약점 관리

1. 무결성 검사
2. 응용 프로그램의 보안 설정 및 패치(Patch) 적용
3. 불필요한 서비스 및 악성 프로그램의 확인과 제거

B 5-4-2 서버 인증

Authentication 인증의 개념

1. 자신의 신원을 시스템에 증명하는 과정
2. 아이디와 패스워드를 입력하는 과정이 가장 일반적인 예시
3. 인증의 주요 유형
    A. 지식 기반 인증 : 주체는 '그가 알고 있는 것'을 보여줌 / 패스워드, PIN
    B. 소유 기반 인증 : 주체는 '그가 가지고 있는 것'을 보여줌 / 토큰 스마트 카드
    C. 생체 기반 인증 : 주체는 '그의 고유한 생체 정보'을 보여줌
    D. 위치 기반 인증 : 주체는 '그의 위치 정보'을 보여줌
    E. 행위 기반 인증 : 주체는 '그가 하는 것'을 보여줌 / 서명, 움직임, 음성
    F. 존재 기반 인증 : 주체는 '그가 본인을 나타내는 것 / 홍채, 지문

B 5-4-3 보안 아키텍처 / 보안 프레임워크

보안 아키텍처

1. 관리적 보안 : ~~정보보호 정책, 정보보호 조직, 정보자산 분류, 정보보호 교육 및 훈련, 인정 보안~~
2. 물리적 보안 : 장비의 파손, 천재지변과 같은 재해의 보안 / 서버 관리실 출입 통제
3. 기술적 보안 : 실행 프로세스 권한 설정, 운영체제 접근 제한, 운영체제의 정보 수집 제한

A 5-4-5 보안 솔루션

침입 탐지 시스템 : IDS

1. ~~오용 탐지~~
2. ~~이상 탐지~~
3. 침입 탐지 시스템의 종류
    A. DMZ
        i. IDS가 설치될 수 있다.
    B. HIDS
        i. 시스템의 내부를 감시하고 분석하는데 중점
        ii. 내부 시스템의 변화를 실시간으로 감시하여 누가 접근, 어떤 작업을 수행 했는지 기록하고 추적
        iii. ~~종류 : OSSEC, md5deep, AIDE, Samhain 등~~
    C. NIDS
        i. ~~외부로부터 침입을 감시하고 분석하는데 중점~~
        ii. 종류 : Snort, Zeek 등

VPN

1. 이용자가 인터넷과 같은 공중망에 사설망을 구축하여 마치 전용망을 사용하는 효과를 가짐

SSH

1. 전소오디는 데이터는 암호화
2. 키를 통한 인증은 클라이언트의 공개키를 서버에 등록
3. 서로 연결되어 있는 컴퓨터 간 원격 명령 실행이나 셸 서비스 등을 수행
4. 기본 네트워크 포트 22번

2기타

정형 기술 검토(FTR)

1. 의제를 제한한다
2. 논쟁과 반박을 제한한다
3. 문제 영역을 명확히 표현한다.

프로그램 설계도 종류

1. NS Chart
    A. 논리의 기술에 중점을 두고 도형을 이용한 표현 방법이다.
    B. 이해하기 쉽고 코드 변환이 용이하다.
    C. 연속, 선택, 반복 등의 제어 논리 구조를 표현한다.

Risk Analysis 리스크 분석

1. 프로젝트에 내재된 위험 요소를 인식하고 그 영향을 분석하여 이를 관리하는 활동
2. 프로젝트를 성공시키기 위하여 위험 요소를 사전에 예측, 대비하는 모든 기술과 활동을 포함

위험 모니터링 - 위험 요소 징후들에 대하여 계속적으로 인지

1. 위험을 이해
2. 첫 번째 조치로 위험을 피할 수 있도록 하는 것
3. 위험 발생 후 즉시 조치

4기타

오류 제어에 사용되는 자동반복 요청방식(ARQ)

1. Stop-and-wait ARQ
2. Go-back-N ARQ
3. Selective-Repeat ARQ
4. ~~Adaptive ARQ~~

모듈의 독립성을 높이기 위한 결합도

1. 오류가 발생했을 때 전파되어 다른 오류의 원인이 되는 파문 효과를 최소화해야 한다.
2. 인터페이스가 정확히 설정되어 있지 않을 경우 불필요한 인터페이스가 모듈 사이의 의존도는 높아지고 결합도가 증가한다.
3. 데이터 교류가 필요한 경우 전역변수보다는 매개변수를 사용하는 것이 결합도를 낮추는 데 도움이 된다.
4. 모듈들이 변수를 공유하여 사용하게 하거나 제어 정보를 교류하게 하면 모듈 간의 결합도가 높아진다

JAVA의 예외(exception)

1. 오동작이나 결과에 악영향을 미칠 수 있는 실행 시간 동안에 발생한 오류
2. 배열의 인덱스가 그 범위를 넘어서는 경우 발생하는 오류
3. 존재하지 않는 파일을 읽으려고 하는 경우에 발생하는 오류

5기타

IPSec 설명

1. ESP : 발신지 인증, 데이터 무결성, 기밀성 모두 보장
2. 운영모드 : Tunnel, Transport 모드로 분류
3. AH : 발신지 호스트를 인증하고 IP 패킷의 무결성을 보장

Stack Guard

메모리상에서 프로그램의 복귀 주소와 변수 사이에 특정 값을 저장해 두었다가

그 값이 변경되었을 경우 오버플로우 상태로 가정하여 프로그램 실행을 중단하는 기술

방화벽 구축 형태

1. Screened Subnet : 외부 네트워크와 내부 네트워크 사이에 완충적인 통신망

보안취약점을 찾는데 사용하는 도구

1. nmap : 서버에 열린 포트 정보를 스캐닝해서 보안취약점을 찾는데 사용하는 도구

정보 보안을 위한 접근 제어

1. 적절한 권한을 가진 인가자만 특정 시스템이나 정보에 접근할 수 있도록 통제하는 것이다.
2. 시스템 및 네트워크에 대한 접근 제어의 가장 기본적인 수단은 IP와 서비스 포트로 볼 수 있다.
3. 네트워크 장비에서 수행하는 IP에 대한 접근 제어로는 관리 인터페이스의 접근제어와 ACL 등 있다.

소프트웨어 개발 프레임워크

1. 재사용성 확대와 성능을 보장 받을 수 있게 하는 개발 소프트웨어
2. 소프트웨어 디자인 패턴을 반제품 소프트웨어 상태로 집적화
3. IoC : 프레임워크의 동작 원리를 그 제어 흐름의 일반적인 프로그램 흐름과 반대로 동작
4. 개발해야 할 애플리케이션의 일부분이 이미 내장된 클래스 라이브러리로 구현되어 있다.
5. 라이브러리를 확장 및 이용
6. JAVA 기반의 대표적인 소프트웨어로는 스프링이 있다.