**2과목 소프트웨어 개발**

1장 데이터 입•출력 구현

036 자료구조

- 자료를 기억장치의 공간 내에 저장하는 방법, 저장된 그룹 내에 존재하는 자료 간의 간계, 처리 방법을 연구 분석하는 것

선형 구조 : 배열 / 선형리스트(연속, 연결 리스트) / 스택 / 큐 / 데크

비선형 구조 : 트리 / 그래프

배열 – 동일한 자료형의 데이터들이 같은 크기로 나열되어 순서를 가지고 있는 집합

선형리스트 – 일정한 순서에 의해 나열된 자료구조

-> 연속리스트 – 배열과 같이 연속되는 기억장소에 저장되는 자료구조

-> 연결리스트 – 순서에 따라 노드의 포인터 부분을 이용하여 서로 연결시킨 자료구조

스택 – 리스트의 한쪽 끝으로만 자료의 삽입, 삭제 작업이 이루어지는 자료 구조 / 후입선출

큐 - 리스트의 한쪽 끝으로만 자료의 삽입 작업이 이루어지고 다른 한쪽에서 삭제 작업이 이루어짐 / 선입선출

037 트리

정점과 선분을 이용하여 사이클을 이루지 않도록 구성한 그래프의 특수한 형태

노드 / 근 노드 / 디그리 / 단말노드 / 자식 노드 / 부모 노드 / 형제 노드 / 트리의 디그리

트리의 운행법

Preorder Inorder Postorder

Root -> Left -> Right Left -> Root -> Right Left -> Right -> Root

수식의 표기법

PreFix InFix PsotFix

연산자 -> Left -> Right Left -> 연산자 -> Right Left -> Right -> 연산자

038 정렬

삽입 정렬 – 이미 순서화된 파일에 새로운 하나의 레코드를 순서에 맞게 삽입시켜 정렬

평균과 최악 모두 수행 시간 복잡도 O()

이미 순서화된 파일에, n번째 키를 앞의 n-1개의 키와 비교

쉘 정렬 – 매개 변수의 값으로 서브파일을 구성하고 서브파일을 삽입 정렬 방식으로 순서 배열하는 과정을 반복하는 정렬 방식

평균 수행 시간 복잡도 O() 최악의 수행 시간 복잡도 O()

매개변수

선택 정렬 - n개의 레코드 중에 최소값을 찾아 첫 번째 레코드 위치에 놓고 나머지 중에서 다시 최솟값을 찾아 두 번째 레코드 위치에 놓는 방식을 반복하여 정렬하는 방식

평균과 최악 모두 수행 시간 복잡도 O()

n개의 레코드 중에서 최소값을 찾아서

버블 정렬 – 인접한 두 개의 레코드 키 값을 비교하여 그 크기에 따라 레코드 위치를 서로 교환하는 정렬 방식

평균과 최악 모두 수행 시간 복잡도 O()

인접한 두 개의 레코드

퀵 정렬 – 하나의 파일을 부분적으로 나누어 가면서 정렬하는 방식

평균 수행 시간 복잡도 O) 최악의 수행 시간 복잡도 O()

하나의 파일을 부분적으로 나누어

힙 정렬 – 전이진 트리를 이용한 정렬 방식

평균과 최악 모두 수행 시간 복잡도 O)

전이진 트리

2-Way 합병 정렬 - 이미 정렬되어 있는 두 개의 파일을 한 개의 파일로 합병하는 정렬방식

평균과 최악 모두 수행 시간 복잡도 O)

이미 정렬된 두 개의 파일을 한 개의 파일로

기수 정렬 – 큐를 이용하여 자릿수별로 정렬하는 방식

평균과 최악 모두 수행 시간 복잡도 O(dn)

버킷

039 검색 – 이분 검색 / 해싱

이분 검색 – 전체 파일을 두 개의 서브파일로 분리해가며 key 레코드를 검색하는 방식

찾고자 하는 key 값을 파일의 중간 레코드 key 값과 비교하면서 검색

중간 레코드 번호 M =

해싱 – 해시 테이블이라는 기억공간을 할당하고 해시함수를 이용하여 레코드 키에 대한 해시 테이블 내의 홈 주소를 계산한 후 주어진 레코드를 해당 기억장소에 저장하거나 검색 작업을 수행하는 방식

해시 테이블 – 버킷 / 슬롯 / collision / Synonym / Overflow

해싱함수 – 제산법 / 제곱법 / 폴딩법 / 기수 변환법 / 대수적 코딩법 / 숫자 분석법 / 무작위법

040 데이터베이스 개요

데이터 저장소 – sw 개발 과정에서 다루어야 할 데이터들을 논리적인 구조로 조직화하거나 물리적인 공간에 구축한 것을 의미

데이터베이스 – 통합된 데이터 / 저장된 데이터 / 운영 데이터 / 공용 데이터

DBMS – 사용자와 데이터베이스 사이에서 사용자의 요구에 따라 정보를 생성해주고 관리해주는 소프트웨어

기능 – 정의 / 조작 / 제어

스키마 – 데이터베이스의 구조와 제약 조건에 관한 전반적인 명세를 기술한 메타데이터의 집합

사용자 관점 - 외부 스키마 / 개념 스키마 / 내부 스키마

041 데이터 입•출력

데이터 입•출력 – 소프트웨어 기능 구현을 위해 데이터베이스에 데이터를 입력하거나 데이터베이스의 데이터를 출력하는 작업을 의미

sql을 사용

데이터 접속 – 개발 코드 내에 SQL 코드를 삽입하거나 객체와 데이터를 연결하는 것

SQL – 국제표준 데이터베이스 언어

데이터 정의어(DDL) / 데이터 조작어(DML) / 데이터 제어어(DCL)

데이터 접속 – 소프트웨어의 기능 구현을 위해 프로그래밍 코드와 데이터베이스의 데이터를 연결하는 것

SQL Mapping / ORM

트랜잭션 – 데이터베이스의 상태를 변환시키는 논리적 기능을 수행하기 위한 작업의 단위

COMMIT / ROLBACK / SAVEPOINT

042 절차형 SQL

프로그래밍 언어와 같이 연속적인 실행이나 분기, 반복 등의 제어가 가능한 SQL

데이터베이스 전용의 간단한 프로그래밍이라고 할 수 있음

프로시저 – 특정기능을 수행하는 일종의 트랜잭션 언어, 호출을 통해 실행되며 저장해 놓은 SQL 작업 수행

트리거 – 데이터베이스 시스템에서 데이터 입력, 갱신, 삭제 등의 이벤트가 발생할 때마다 관련 작업이 자동으로 수행

사용자 정의 함수 – 프로시저와 유사하게 SQL을 사용하여 일련의 작업을 연속적으로 처리, 종료시 결과를 단일값으로 반환

디버깅을 통해 기능의 적합성 여부를 검증하고 실행을 통해 결과를 확인하는 테스트 과정 수행

쿼리 성능 최적화 – 데이터 입•출력 애플리케이션의 성능 향상을 위해 SQL 코드를 최적화하는 것

문제정리

자료 구조 분류

선형 구조 : 배열 , 스택, 큐, 테크, 선형 리스트

비선형 구조 : 트리, 그래프

PUSH – 스택에 자료 입력하는 명령

POP – 스택에서 자료를 출력하는 명령

오버플로 – 큐가 꽉 채워져 있는 상태로 자료를 삽입 할 수 없는 상태

언더플로 – 자료가 없어서 자료를 제거할 수 없는 상태

선택 정렬 - 1단계를 마치면 가장 작은 값이 맨 앞으로 오고 2단계를 마치면 두 번째로 작은 값이 두 번째에 옴

버블 정렬 - 1단계를 마치면 가장 큰 값이 맨 뒤로 가고 2단계를 마치면 두 번째로 큰 값이 뒤에서 두 번째에 위치함

삽입 정렬 - 1단계를 마치면 첫 번째와 두 번째 값만 비교하여 교환

선형 검색 – 정렬 되어 있지 않은 파일에서 순차적으로 검색하는 방식, KEY값을 이용하여 검색하는 방법

통합된 데이터 – 자료의 중복을 배제한 데이터의 모임

저장된 데이터 – 컴퓨터가 접근할 수 있는 저장 매체에 저장된 자료

운영 데이터 – 조직의 고유한 업무를 수행하는 데 존재 가치가 확실하고 없어서는 안 될 반드시 필요한 자료

공용 데이터 – 응용 시스템들이 공동으로 소유하고 유지하는 자료

깊이 우선 탐색 – 정점에서 자식 노드 방향으로 운행하면서 형제 노드와 자식 노드가 있을 때 자식 노드를 우선 탐색하는 기법 모든 노드를 한번씩 방문함

TCL : COMMIT – 변경 내용을 반영 / ROLLBACK – 이전 상태를 되돌림 / SAVEPOINT – 중간 저장점을 지정

키워드 : 1. 배열 2. 스택 3. 그래프 4. 트리 5. 정렬 6. 데이터베이스 7. DBMS 8. 스키마 9. SQL 10. 트랜잭션

2장 통합 구현

043 단위 모듈 구현

단위 모듈 – 소프트웨어 구현에 필요한 여러 동작 중 한 가지 동작을 수행하는 기능을 모듈로 구현한 것

단위 기능 명세서 작성 -> 입•출력 기능 구현 -> 알고리즘 구현

단위 기능 명세서 – 설계서를 작성하는 기능 및 코드 명세서나 설계 지침과 같이 단위 기능을 명세화한 문서들을 의미

입•출력 기능 구현 – 단위 기능 명세서에 정의한 데이터 형식에 따라 입•출력 기능을 위한 알고리즘 및 데이터를 구현

알고리즘 구현 – 입•출력 데이터를 바탕으로 단위 기능별 요구 사항들을 구현 가능한 언어를 이용하여 모듈로 구현

디바이스 드라이버 모듈 / 네트워크 모듈 / 파일 모듈 / 메모리 모듈 / 프로세스 모듈

044 단위 모듈 테스트

- 프로젝트의 단위 기능을 구현하는 모듈이 정해진 기능을 정확히 수행하는 지 검증하는 것

화이트 박스 테스트 / 블랙박스 테스트

테스트 케이스 – 구현된 소프트웨어가 사용자의 요구사항을 정확하게 준수했는지를 확인하기 위해 설계된 입력값, 실행 조건, 기대 결과로 구성된 테스트 항목에 대한 명세서

식별자 / 테스트 항목 / 입력 명세 / 출력 명세 / 환경 설정 / 특수 절차 요구 / 의존성 기술

테스트 프로세스 – 테스트를 위해 수행하는 모든 작업들이 테스트의 목적과 조건을 달성할 수 있도록 도와주는 과정

테스트 프로세스 5단계

1. 계획 및 제어 단계

2. 분석 및 설계 단계

3. 구현 및 실현 단계

4. 평가 단계

5. 완료 단계

045 개발 지원 도구

통합 개발 환경 – 모든 작업을 하나의 프로그램에서 처리할 수 있도록 제공하는 소프트웨어적인 개발 활동

코딩 / 컴파일 / 디버깅 / 배포

빌드 – 소스 코드 파일들을 컴퓨터에서 실행할 수 있는 제품 소프트웨어로 변환하는 과정 또는 결과물

Ant / Maven / Gradle

협업 도구 – 개발에 참여하는 사람들이 서로 다른 작업 환경에서 원활히 프로젝트를 수행할 수 있도록 도와주는 도구

프로젝트 및 일정 관리 / 정보 공유 및 커뮤니케이션 / 디자인

문제정리

컴포넌트 – 독립적인 업무 또는 기능을 수행하는 단위, 실행 코드 기반으로 작성된 모듈

인터페이스 – 서로 다른 두 시스템이나 소프트웨어 등을 서로 이어주는 부분, 접속 장치

라이브러리 – 개발 편의를 위해 자주 사용되는 코드, API, 클래스, 값, 자료형 등의 다양한 자원들을 모아 놓은 것

Ant – 의존성을 사용하지 않고 직접 라이브러리를 다운받거나 연결시켜서 사용함

Maven – 의존성을 설정하여 라이브러리 관리하는 빌드 도구

공유 메모리 – 공유 가능한 메모리를 구성하여 다수의 프로세스 간 통신을 지원

소켓 – 네트워크 소켓을 이용한 프로세스 간 통신을 지원

세마포어 – 자원에 대한 접근 제어를 통해 프로세스 간 통신을 지원

메시지 큐잉 – 메시지 전달 방식을 사용하여 프로세스 간 통신을 지원

단위 테스트를 수행하기 위해서는 테스트 수행 전에 단독 실행을 위한 환경과 테스트 데이터를 준비해야 함

테스트 케이스 – 테스트에 필요한 입력 데이터, 테스트 조건, 예상 결과 등을 기입한 문서

- 테스트 수행에 필요한 환경이나 테스트 케이스 간의 의존성을 기입한 문서

테스트 시나리오 – 여러 개의 테스트 케이스를 묶은 것으로 테스트 케이스 수행에 대한 절차를 명세한 문서

테스트 프로시저 – 테스트 스크립트라 불리며, 테스트 케이스의 실행 순서 의미

기능별 구현 모듈

• 디바이스 드라이버 모듈 : 하드웨어 주변 장치의 동작을 구현한 모듈

• 네트워크 모듈 : 네트워크 장비 및 데이터 통신을 위한 기능을 구현한 모듈

• 파일 모듈 : 컴퓨터 내부의 데이터 구조 영역에 접근하는 방법을 구현한 모듈

• 메모리 모듈 : 파일을 프로세스의 가상 메모리에 매핑/해제하는 방법과 프로세스 사이의 통신 기능을 구현한 모듈

• 프로세스 모듈 : 하나의 프로세스 안에서 다른 프로세스를 생성하는 방법을 구현한 모듈

테스트 프로세스 5단계

• 계획 및 제어 : 테스트 목표를 달성하기 위한 계획을 수립하고 계획대로 진행되도록 제어하는 단계

• 분석 및 설계 : 테스트 목표를 구체화하여 테스트 시나리오와 테스트 케이스를 작성하는 단계

• 구현 및 실현 : 효율적인 테스트 수행을 위해 테스트 케이스들을 조합하여 테스트 프로시저에 명세하는 단계

• 평가 : 테스트가 계획과 목표에 맞게 수행되었는지 평가하고 기록하는 단계

• 완료 : 이후의 테스트를 위한 참고 자료 및 테스트 수행에 대한 증거 자료로 활용하기 위해 수행 과정과 산출물을 기록 및 저장하는 단계

키워드 : 1. 단위 기능 명세서 2. IPC 3. 테스트 케이스 4. 단위 모듈 테스트 5. 테스트 프로세스 6. IDE 7. 빌드 도구 8. 단위 모듈 9. Ant 10. Maven

3장 제품 소프트웨어 패키징

046 소프트웨어 패키징

- 모듈별로 생성한 실행파일을 묶어 배포용 설치 파일을 만드는 것

패키징 작업 순서

기능 식별 -> 모듈화 -> 빌드 진행 -> 사용자 환경 분석 -> 패키징 및 적용 시험 -> 패키징 변경 개선 -> 배포

패키징 주기 – 애자일 기법인 경우 2~4주 / 각 주기가 끝날 때마다 패키징을 수행

온라인 배포 / 오프라인 배포

047 릴리즈 노트 작성

- 개발 과정에서 정리된 릴리즈 정보를 소프트웨어의 최종 사용자인 고객과 공유하기 위한 문서

릴리즈 노트는 현재 시제로 작성해야 함 / 신규 소스, 빌드 이력이 정확하게 관리되어야 함 / 표준형식은 X 보통 머리말, 개요, 목적, 문제 요약, 재현 항목, 수정/개선 내용, 사용자 영향도, SW 지원 영향도, 노트, 면책 조항, 연락처가 포함됨

릴리즈 노트 작성 순서

모듈 식별 -> 릴리즈 정보 확인 -> 릴리즈 노트 개요 작성 -> 영향도 체크 -> 정식 릴리즈 노트작성 -> 추가 개선 항목 식별

048 디지털 저작권 관리(DRM)

저작권 – 저작물에 대하여 창작자가 가지는 배타적 독점적 권리로 타인의 침해를 받지 않을 고유한 권한

디지털 저작권 관리 – 저작권자가 배포한 디자털 콘텐츠가 저작권자가 의도한 용도로만 사용되도록 전 과정에 걸쳐 사용되는 디지털 콘텐츠 관리 및 보호 기술

디지털 저작권 관리의 흐름 및 구성 요소

클리어링 하우스 – 권한 정책 / 라이선스

콘텐츠 제공자 – 패키저 / 콘텐츠 / 메타 데이터

콘텐츠 분배자 – 유통 시스템

콘텐츠 소비자 – DRM 컨트롤러 / 보안 컨테이너

디지털 저작권 관리 기술 – 암호화 / 키 관리 / 암호화 파일 생성 / 식별 기술 / 저작권 표현 / 정책 관리 / 크랙 방지 / 인증

049 소프트웨어 설치 매뉴얼 작성

- 개발 초기부터 적용된 기준이나 사용자가 소프트웨어를 설치하는 과정에 필요한 내용을 기록한 설명서와 안내서

서문 – 문서 이력, 설치 매뉴얼의 주석, 설치 도구의 구성, 설치 환경 체크 항목을 기술

기본 사항 – 소프트웨어 개요 / 설치 관련 파일 / 설치 아이콘 / 프로그램 삭제 / 관련 추가 정보

설치 매뉴얼 – 사용자가 설치 과정을 이해하기 쉽도록 설치 화면을 누락 없이 캡처하고 순서대로 상세히 설명

설치 화면 및 UI – 설치 실행, 메인 화면 및 안내창 / 설치 이상 메시지 설명 / 설치 완료 및 결과 / FAQ / 설치 시 점검 사항 / Network 환경 및 보안 / 고객 지원 방법 / 준수 정보 & 제한 보증

설치 매뉴얼 작성 순서

기능 식별 -> UI 분류 -> 설치 파일/백업 파일 확인 -> Uninstall 절차 확인 -> 이상 Case 확인 -> 최종 매뉴얼 적용

050 소프트웨어 사용자 매뉴얼 작성

- 소프트웨어를 사용하는 과정에서 필요한 내용을 문서로 기록한 설명서와 안내서

서문 – 문서 이력, 사용자 매뉴얼의 주석, 기록 보관을 위해 필요한 내용 기술

문서 이력 : 버전 / 작성자 / 작성일 / 검토자 / 일시 / 검수인

사용자 매뉴얼의 주석 : 주의 사항 / 참고 사항

기록 보관 애용 : 필요한 기술 지원이나 추가 정보를 얻기 위한 소프트웨어 등록 정보를 기술

소프트웨어 명칭 / 모델명 / 문서 번호 / 제품 번호 / 구입 날짜

기본 사항 – 소프트웨어 개요, 소프트웨어 사용 환경, 소프트웨어 관리, 모델, 버전별 특징, 기능, 인터페이스의 특징, 소프트웨어 구동 환경

사용자 매뉴얼 : 사용자가 사용 방법을 이해하기 쉽도록 상황별로 누락 없이 캡처하여 순서대로 상세히 설명

사용자 화면 및 UI / 주요 기능 분류 / 응용 프로그램 및 설정 / 장치 연동 / Network 환경 / Profile 안내, 고객 치원 방법, 준수 정보 및 제한 보증

사용자 매뉴얼 작성 순서 : 작성 지침 정의 -> 사용자 매뉴얼 구성 요소 정의 -> 구성 요소별 내용 작성 -> 사용자 매뉴얼 검토

051 소프트웨어 버전 등록

소프트웨어 패키징의 형상 관리 – 소프트웨어의 개발 과정에서 변경 사항을 관리하기 위해 개발된 일련의 활동

형상 관리 기능 : 항상 식별 / 버전 제어 / 형상 통제(변경 관리) / 형상 감사 / 형상 기록

sw 버전 등록 관련 주요 기능 : 저장소 / 가져오기 / 체크아웃 / 체크인 / 커밋 / 동기화

소프트웨어 버전 등록 과정 : 가져오기 -> 인출 -> 예치 -> 동기화 -> 차이

052 소프트웨어 버전 관리 도구

공유 폴더 방식 – 버전 관리 자료가 로컬 컴퓨터의 공유 폴더에 저장되어 관리 되는 방식

SOCS / RCS / PVCS / QVCS

클라이언트/서버 방식 – 버전 관리 자료가 중앙 시스템에 저장되어 관리되는 방식

CVS / SVN / CVSNT / Clear Case / CMVC / Perfore

분산 저장소 방식 – 버전 관리 자료가 하나의 원격 저장소와 분산된 개발자 PC의 로컬 저장소에 함께 저장되어 관리되는 방식

Git / GNU arch / DCVS / Bazaar / Mercurial / TeamWare / Bitkeeper / Plastic SCM

Subversion – CVS를 개선한 것으로 아파치 소프트웨어 재단에서 2000년도에 발표

add / commit / update / checkout / lock/unlock / import / export / info / diff / merge

Git – 리눅스 커널 개발에 사용할 관리 도구로 개발

add / commit / branch / checkout / merge / init / remote add / push / fetch

053 빌드 자동화 도구

빌드 – 소스 코드 파일들을 컴파일한 후 여러 개의 모듈을 묶어 실행 파일로 만드는 과정

이러한 빌드를 포함하여 테스트 및 배포를 자동화하는 도구 – 빌드 자동화 도구

Ant / Make / Maven / Gradle / Jenkins

Jenkins – JAVA 기반 오픈 소스 형태

Gradle – Groovy를 기반한 오픈 소스 형태, 안드로이드 앱 개발 환경에서 사용

문제 정리

패키징은 각 주기가 끝날 때마다 수행

모듈화 과정 – 확인된 기능 단위로 코드 분류

빌드 진행 과정 – 모듈 단위별로 실행 파일 만듦

디지털 저작권 관리 – 아날로그인 경우 디지털로 변환한 후 패키지에 의해 DRM 패키징을 수행

기술 요소 – 암호화, 키 관리, 암호화 파일 생성, 식별 기술, 저작권 표현 정책 관리, 크랙 방지. 인증

최종 매뉴얼 적용 단계 – 문의 답변을 정리하여 기록하고 완성된 매뉴얼을 검토한 후 고객 지원에 대한 내용 기록

init – 지역 저장소 생성 checkout – 지정한 브랜치로 이동

형상 통제 – 식별된 형상 항목에 대한 변경 요구를 검토하여 기준선이 반영될 수 있도록 조정

형상 관리 – 소프트웨어의 변경 사항을 관리하기 위해 개발된 활동

콘텐츠 제공자 – 콘텐츠를 제공하는 저작권자

패키저 – 콘텐츠를 메타 데이터와 함께 배포 가능한 단위로 묶음

클리어링 하우스 – 라이선스를 발급하고 관리

DRM 컨트롤러 – 배포된 콘텐츠의 이용 권한 통제

키워드 : 1. 소프트웨어 패키징 2. 릴리지 노트 3. DRM 4. 소프트웨어 설치 매뉴얼 5. 소프트웨어 사용자 매뉴얼 6. 형상 관리 7. Subversion 8. Git 9. Jenkins 10. Gradle

4장 애플리케이션 테스트 관리

054 애플리케이션 테스트

- 애플리케이션에 잠재되어 있는 결함을 찾아내는 일련의 행위 또는 절차

- 개발된 소프트웨어가 고객의 요구사항을 만족시키는지 확인하고 소프트웨어가 기능을 정확히 수행하는지 검증함

필요성

– 프로그램 실행 전 오류를 발견하여 예방할 수 있음

- 신뢰도를 향상시킴

- 새로운 오류의 유입도 예방 가능

- 최소한의 시간과 노력으로 많은 결과를 찾을 수 있음

055 애플리케이션 테스트의 분류

프로그램 실행 여부에 따른 테스트 – 정적 테스트 / 동적 테스트

테스트 기반에 따른 테스트 – 명세 기반 테스트 / 구조 기반 테스트 / 경험 기반 테스트

시각에 따른 테스트 – 검증 테스트 / 확인 테스트

목적에 따른 테스트 – 회복 테스트 / 안전 테스트 / 강도 테스트 / 성능 테스트 / 구조 테스트 / 회귀 테스트 / 병행 테스트

056 테스트 기법에 따른 애플리케이션 테스트

화이트박스 테스트

- 모듈의 원시 코드를 오픈시킨 상태에서 원시 코드의 논리적인 모든 경로를 테스트하여 테스트 케이스를 설계하는 방법

종류 – 기초 경로 검사 / 제어 구조 검사

검증 기준 – 문장 검증 기준 / 분기 검증 기준 / 조건 검증 기준 / 분기 검증 기준

블랙박스 테스트

- 소프트웨어가 수행할 특정 기능을 알기 위해서 각 기능이 완전히 작동되는 것을 입증하는 테스트로 기능 테스트라고도 함

종류 – 동치 분할 검사 / 경계값 분석 / 원인-효과 그래프 검사 / 오류 예측 검사 / 비교 검사

057 개발 단계에 따른 애플리케이션 테스트

- 소프트웨어 개발 단계에 따라 단위 테스트, 통합 테스트, 시스템 테스트, 인수 테스트로 분류

- 소프트웨어의 개발단계에서부터 테스트를 수행하므로 단순히 소프트웨어에 포함된 코드 상의 오류뿐만 아니라 요구 분석의 오류, 설계 인터페이스 오류 등도 발견할 수 있음

단위 테스트 – 코딩 직후 소프트웨어 설계의 최소 단위인 모듈이나 컴포넌트에 초점을 맞춰 테스트 하는 것

통합 테스트 – 단위 테스트가 완료된 모듈들을 결합하여 하나의 시스템으로 완성시키는 과정에서의 테스트

시스템 테스트 – 개발된 소프트웨어가 해당 컴퓨터 시스템에서 완벽학 수행되는가를 점검하는 테스트

인수 테스트 – 개발한 소프트웨어가 사용자의 요구사항을 충족하는지에 중점을 두고 테스트

058 통합 테스트

- 단위 테스트가 끝난 모듈을 통합하는 과정에서 발생하는 오류 및 결함을 찾는 테스트 기법

비점진적 통합 방식 / 점진적 통합 방식

하향식 통합 테스트 – 프로그램의 상위 모듈에서 하위 모듈 방향으로 통합하면서 테스트하는 기법

1. 주요 제어 모듈은 작성된 프로그램을 사용하고 주요 제어 모듈의 종속 모듈은 스텁으로 대체

2. 깊이 우선 또는 넓이 우선 등의 통합 방식에 따라 하위 모듈인 스텁들이 한 번에 하나씩 실제 모듈로 교체됨

3. 모듈이 통합될 때마다 테스트를 실시

4. 새로운 오류가 발생하지 않음을 보증하기 위해 희귀 테스트를 실시

상향식 통합 테스트 – 프로그램의 하위 모듈에서 상위 모듈 방향으로 통합하면서 테스트하는 기법

1. 하위 모듈들을 클러스터로 결합

2. 상위 모듈에서 데이터의 입,출력을 확인하기 위해 더미 모듈인 드라이버를 작성

3. 통합된 클러스터 단위로 테스트

4. 테스트가 완료되면 클러스터는 프로그램 구조의 상위로 이동하여 결합하고 드라이버는 실제 모듈로 대체

혼합식 통합 테스트 – 하위 수준에서는 상향식 통합, 상위 수준에서는 하향식 통합을 사용하여 최적의 테스트를 지원하는 방식 / 샌드위치식 통합 테스트 방법

회귀 테스팅 – 이미 테스트된 프로그램의 테스팅을 반복하는 것 / 통합 테스트로 인해 변경된 모듈이나 컴포넌트에 새로운 오류가 있는 지 확인하는 테스트

059 애플리케이션 테스트 프로세스

- 개발된 소프트웨어가 사용자의 요구대로 만들어졌는지, 결함은 없는지 등을 테스트하는 절차

테스트 계획 -> 테스트 분석 및 디자인 -> 테스트 케이스 및 시나리오 작성 -> 테스트 수행 -> 테스트 결과 평가 및 리포팅 -> 결함 추적 및 관리

테스트 계획 – 프로젝트 계획석, 요구 명세서 등을 기반으로 테스트 목표를 정의하고 테스트 대상 및 범위를 결정

테스트 분석 및 디자인 – 테스트의 목적과 원칙을 검토하고 사용자의 요구사항을 분석

테스트 케이스 및 시나리오 작성 – 테스트 케이스의 설계 기법에 따라 테스트 케이스를 작성하고 검토 및 확인한 후 테스트 시나리오를 작성

테스트 수행 – 테스트 환경을 구축한 후 테스트 수행

테스트 결과 평가 및 리포팅 – 테스트 결과를 비교 분석하여 테스트 결과서를 작성

결함 추적 및 관리 – 테스트를 수행한 후 결함이 어디에서 발생했는지, 어떤 종류의 결함인지 등 결함을 추적하고 관리

060 테스테 케이스 / 테스트 시나리오 / 테스트 오라클

테스트 케이스 – 구현된 소프트웨어가 사용자의 요구사항을 정확하게 준수했는지를 확인하기 위해 설계된 입력 값, 실행 조건, 기대 결과 등으로 구성된 테스트 항목에 대한 명세서

테스트 계획 검토 및 자료 확보 -> 위험 평가 및 우선순위 결정 -> 테스트 요구사항 정의 -> 테스트 구조 설계 및 테스트 방법 결정 -> 테스트 케이스 정의 -> 테스트 케이스 타당성 확인 및 유지 보수

테스트 시나리오 – 테스트 케이스를 적용하는 순서에 따라 여러 개의 테스트 케이스들을 묶은 집합으로 테스트 케이스들을 적용하는 구체적인 절차를 명세한 문서

테스트 오라클 – 테스트 결과가 올바른지 판단하기 위해 사전에 정의된 참 값을 대입하여 비교하는 기법 및 활동을 말함

참 오라클 / 샘플링 오라클 / 추정 오라클 / 일관성 오라클

061 테스트 자동화 도구

테스트 자동화 – 사람이 반복적으로 수행하던 테스트 절차를 스크립트 형태로 구현하는 자동화 도구를 적용함으로써 쉽고 효율적으로 테스트를 수행할 수 있도록 한 것

테스트 자동화 도구 유형

정적 분석 도구 / 테스트 케이스 생성 도구 / 테스트 실행 도구 / 성능 테스트 도구 / 테스트 통제 도구 / 테스트 하네스 도구

테스트 수행 단계별 테스트 자동화 도구

테스트 계획 -> 요구사항 관리

테스트 분석/설계 -> 테스트 케이스 생성

테스트 수행 -> 테스트 자동화 / 정적 분석 / 동적 분석 / 성능 테스트 / 모니터링

테스트 관리 -> 커버리지 분석 / 형상 관리 / 결함 추적/관리

062 결함 관리

결함 – 오류 발생, 작동 실패 등 소프트웨어가 개발자가 설계한 것과 다르게 동작하거나 다른 결과가 발생되는 것을 의미

결함 관리 계획 -> 결함 기록 -> 결함 검토 -> 결함 수정 -> 결함 재확인 -> 결함 상태 추적 및 모니터링 활동 -> 최종 결함 분석 및 보고서 작성

결함 상태 추적 – 결함 분포 / 결함 추세 / 결함 에이징

결함 추적 순서 : 결함 등록 -> 결함 검토 -> 결함 할당 -> 결함 수정 -> 결함 조치 보류 -> 결함 종료 -> 결함 해제

결함 분류 – 시스템 결함 / 기능 결함 / GUI 결함 / 문서 결함

결함 심각도 – High / Medium / Low

결함 관리 도구 – Maintis / Trac / Redmine / Bugzilla

063 애플리케이션 성능 분석

애플리케이션 성능 – 사용자가 요구한 기능을 최소한의 자원을 사용하여 최대한 많은 기능을 신속하게 처리하는 정도

측정 지표 – 처리량 / 응답 시간 / 경과 시간 / 자원 사용률

성능 테스트 도구 – 애플리케이션의 성능을 테스트하기 위해 애플리케이션에 부하나 스트레스를 가하면서 애플리케이션의 성능 측정 지표를 점검하는 도구

JMeter / LoadUI / OpenSTA

시스템 모니터링 도구 – 애플리케이션이 실행되었을 때 시스템 자원의 사용량을 확인하고 분석하는 도구

Scouter / Zabbix

064 복잡도

- 시스템이나 시스템 구성 요소 또는 소프트웨어의 복잡한 정도를 나타나는 말

시간 복잡도 – 알고리즘의 실행시간, 알고리즘을 수행하기 위해 프로세스가 수행하는 연산 횟수를 수치화한 것을 의미

빅오 표기법 / 세타 표기법 / 오메가 표기법

빅오 표기법 – 알고리즘의 실행시간이 최악일 때 표기하는 방법

O(1) / O) / O(n) / O) / O() / O()

순한 복잡도 – 한 프로그램의 논리적인 복잡도를 측정하기 위한 소프트웨어 척도

065 애플리케이션 성능 개선

소스코드 최적화 – 나쁜 코드를 배제하고 클린 코드로 작성하는 것

클린 코드 / 나쁜 코드 – 스파게티 코드 / 외계인 코드

클린 코드 작성 원칙 – 가독성 / 단순성 / 의존성 배제 / 중복성 최소화 / 추상화

소스 코드 최적화 유형 – 클래스 분할 배치 / 느슨한 결합 / 코딩 형식 준수 / 좋은 이름 사용 / 적절한 주석문 사용

소스 코드 품질 분석 도구 – 소스 코드의 코딩 스타일, 코드에 설정된 코딩 표준, 코드의 복잡도, 코드에 존재하는 메모리 누수 현상, 스레드 결함 등을 발견하기 위해 사용하는 분석 도구

정적 분석 도구 / 동적 분석 도구

소스 코드 품질 분석 도구 종류

pmd / cppcheck / SonarQube / checkstyle / ccm / cobertura / Avalanche / Valgrind

5장 인터페이스 구현

066 모듈 간 공통 기능 및 데이터 인터페이스 확인

공통 기능 – 모듈의 기능 중에서 공통적으로 제공되는 기능을 의미

데이터 인터페이스 – 모듈 간 교환되는 데이터가 저장될 파라미터를 의미

모듈 간 공통 기능 및 데이터 인터페이스 확인 순서

1. 인터페이스 설계서를 통해 모듈별 기능을 확인

2. 외부 및 내부 모듈을 기반으로 공통적으로 제공되는 기능과 각 데이터의 인터페이스를 확인

인터페이스 설계서 – 시스템 사이의 데이터 교환 및 처리를 위해 교환 데이터 및 관련 업무, 송•수신, 시스템 등에 대한 내용을 정의한 문서

일반적인 인터페이스 설계서 – 시스템의 인터페이스 목록, 각 인터페이스의 상세 데이터 명세, 각 기능의 세부 인터페이스 정보를 정의한 문서

시스템 인터페이스 설계서 / 상세 기능별 인터페이스 명세서

정적•동적 모형을 통한 인터페이스 설계서 – 정적•동적 모형으로 각 시스템의 구성 요소를 표현한 다이어그램을 이용하여 만든 문서

인터페이스 설계서에서 정의한 모듈을 기반으로 각 모듈의 기능을 확인

067 모듈 연계를 위한 인터페이스 기능 식별

모듈 연계 개요 – 내부 모듈과 외부 모듈 또는 내부 모듈 간 데이터의 교환을 위해 관계를 결정하는 것

EAI – 기업 내 각종 애플리케이션 및 플랫폼 간의 정보 전달, 연계, 통합 등 상호 연동이 가능하게 해주는 솔루션

EBS – 애플리케이션 간 연계, 데이터 변환, 웹 서비스 지원 등 표준 기반의 인터페이스를 제공하는 솔루션

068 모듈 간 인터페이스 데이터 표준 확인

인터페이스 데이터 표준 – 모듈 간 인터페이스에 사용되는 데이터의 형식을 표준화한 것

모듈 간 인터페이스 데이터 표준 확인 순서

1. 데이터 인터페이스를 통해 인터페이스 데이터 표준을 확인

2. 인터페이스 기능을 통해 인터페이스 표준을 확인

3. 데이터 인터페이스와 인터페이스 기능을 통해 확인된 인터페이스 표준을 검토하여 최종적인 인터페이스 데이터 표준을 확인

데이터 인터페이스 확인 – 식별된 데이터 인터페이스에서 입•출력값의 의미와 데이터의 특성을 구체적으로 확인

인터페이스 기능 확인 – 식별된 인터페이스 기능을 기반으로 인터페이스 기능 구현을 위해 필요한 데이터 항목을 확인

인터페이스 데이터 표준 확인 – 데이터 인터페이스에서 확인된 데이터 표준과 인터페이스 기능을 통해 확인된 데이터 항목들을 검토하여 최종적으로 데이터 표준 확인

069 인터페이스 기능 구현 정의

- 인터페이스를 실제로 구현하기 위해 인터페이스 기능에 대한 구현 방법을 기능별로 기술한 것

인터페이스 기능 구현 정의 순서

1. 컴포넌트 명세서를 확인

2. 인터페이스 명세서를 확인

3. 일관된 인터페이스 기능 구현을 정의

4. 정의된 인터페이스 기능 구현을 정형화

모듈 세부 설계서 – 모듈의 구성 요소와 세부적인 동작 등을 정의한 설계서

컴포넌트 명세서 -> 컴포넌트의 개요 및 내부 클래스의 동작, 인터페이스를 통해 외부와 통신하는 명세 등을 정의한 것

인터페이스 명세서 -> 컴포넌트 명세서의 항목 중 인터페이스 클래스의 세부 조건 및 기능 등을 정의한 것

모듈 세부 설계서 확인 – 각 모듈의 컴포넌트 명세서와 인터페이스 명세서를 기반으로 인터페이스에 필요한 기능을 확인

인터페이스 기능 구현 정의 – 인터페이스의 기능, 인터페이스 데이터 표준, 모듈 세부 설계서를 기반으로 일관성 있고 정형화된 인터페이스 기능 구현에 대해 정의

070 인터페이스 구현

- 송•수신 시스템 간의 데이터 교환 및 처리를 실현해 주는 작업을 의미

데이터 통신을 이용한 인터페이스 구현 – 애플리케이션 영역에서 인터페이스 형식에 맞춘 데이터 포맷을 인터페이스 대상으로 전송하고 이를 수신 측에서 파싱하여 해석하는 방법

인터페이스 엔티티를 이용한 인터페이스 구현 – 인터페이스가 필요한 시스템 사이에 별도의 인터페이스 엔티티를 두어 상호 연계하는 방법

071 인터페이스 예외 처리

- 구현된 인터페이스가 동작하는 과정에서 기능상 예외 상황이 발생 했을 때 이를 처리하는 절차

데이터 통신을 이용한 인터페이스 예외 처리 – 인터페이스 객체를 이용해 구현한 인터페이스 동작이 실패할 경우를 대비한 것으로 인터페이스 객체의 송•수신 시 발생할 수 있는 예외 케이스를 정의하고 각 예외 케이스마다 예외 처리 방법을 기술

인터페이스 엔티티를 이용한 인터페이스 예외 처리 – 인터페이스 동작이 실패할 경우를 대비하여 해당 엔티티에 인터페이스의 실패 상황과 원인 등을 기록하고 이에 대한 조치를 취할 수 있도록 사용자 및 관리자에서 알려주는 방식으로 예외 처리 방법을 정의

072 인터페이스 보안

인터페이스는 시스템 모듈 간 통신 및 정보 교환을 위한 통로로 사용되므로 충분한 보안 기능을 갖추지 않으면 시스템 모듈 전체에 악영향을 주는 보안 취약점이 될 수 있음

인터페이스 기능이 수행되는 각 구간들의 구현 현황을 확인하고 각 구간에 어떤 보안 취약점이 있는지를 분석함

분석한 인터페이스 기능과 보안 취약점을 기반으로 인터페이스 보안 기능을 적용함

-> 네트워크 영역 / 애플리케이션 영역 / 데이터베이스 영역

데이터 무결성 검사 도구는 시스템 파일의 변경 유무를 확인하고 파일이 변경되었을 경우 이를 관리자에게 알려주는 도구로 인터페이스 보안 취약점을 분석하는데 사용됨

073 연계 테스트

- 구축된 연계 시스템과 연계 시스템의 구성 요소가 정상적으로 동작하는지 확인하는 활동

연계 테스트 케이스 작성 – 연계 시스템 간의 데이터 및 프로세스의 흐름을 분석하여 필요한 테스트 항목을 도출하는 과정

-> 송•수신용 연계 응용 프로그램의 단위 테스트 케이스

-> 연계 테스트 케이스

연계 테스트 환경 구축 – 테스트의 일정, 방법, 절차, 소요 시간 등을 송•수신 기관과 협의를 통해 결정

연계 테스트 수행 – 연계 응용 프로그램을 실행하여 연계 테스트 케이스의 시험 항목 및 처리 절차 등을 실제로 진행하는 것

연계 테스트 수행 결과 검증 – 연계 테스트 케이스의 시험 항목 및 처리 절차를 수행한 결과가 예상 결과와 동일한지를 확인하는 것

074 인터페이스 구현 검증

- 인터페이스가 정상적으로 문제없이 작동하는지 확인하는 것

인터페이스 구현 검증 도구

xUnit / STAF / FitNesse / NTAF / Selenium / water

인터페이스 구현 감시 도구

스카우터 / 제니퍼

- 인터페이스 기능 구현 정의를 통해 구현된 인터페이스 명세서의 세부 기능을 참조하여 인터페이스의 정상적인 동작 여부를 확인하기 위한 검증 도구와 감시 도구의 요건을 분석