\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

36주차 - 자료 구조

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 자료 구조의 정의

1) 저장 공간의 효율성, 실행시간의 신속성을 고려해 기억장치 내 저장 방법, 자료 간 관계 처리 방법 연구

2) 자료의 표현과 관련된 연산

3) 일련의 자료들을 조직 및 구조화

4) 모든 자료구조에서는 필요한 모든 연산 처리 가능

2. 자료 구조의 분류

1) 선형 구조

- 배열

- 선형 리스트

- 연속 리스트

- 연결 리스트

- 스택

- 큐

- 데크

2) 비선형 구조

- 트리

- 그래프

3. 배열

1) 동일한 자료형의 데이터들이 같은 크기로 나열되어 순서를 갖는 집합

2) 정적인 자료 구조, 기억장소 추가 어려움, 데이터 삭제 시 빈 공간 생겨 낭비 발생

3) 첨자 이용해 접근, 반복적 데이터 처리 작업

4) ex) a[1][1];

4. 선형 리스트

1) 연속 리스트

- 배열을 이용

- 연속되는 기억장소에 저장됨

- 기억장소를 연속적으로 배정받아 기억장소 이용 효율은 밀도가 1로서 가장 좋다.

- 중간에 데이터 삽입,삭제 시 자료의 이동 필요

2) 연결 리스트

- 포인터를 이용

- 임의의 기억공간에 기억시켜 자료 항목 순서에 따라 노드의 포인터 부분을 이용해 서로 연결

- 노드 : 데이터부분, 링크부분으로 이루어짐

- 노드의 삽입, 삭제가 쉬움

- 포인터 부분이 필요하기 때문에 기억공간 이용 효율이 낮음

- 포인터 찾는 시간이 걸려 접근 속도가 느림

- 중간 노드 연결이 끊어지면 다음 노드 찾기 어려움

5. 스택

1) 한쪽 끝으로만 삽입,삭제 (LIFO : Last In First Out)

2) 함수 호출 순서 제어, 인터럽트 처리, 수식 계산 및 수식 표기법 등에 사용

3) 공간이 가득 차있을 때 삽입하면 오버플로, 공간이 비어있을 때 삭제하면 언더플로

- Top : 가장 마지막에 삽입된 데이터의 위치

- Bottom : 가장 아래

- Push : 자료의 삽입

- Pop : 자료의 삭제

6. 큐(Queue)

1) 한쪽은 입력 다른 쪽은 출력 (FIFO : First In First Out)

2) 두 개의 포인터가 있음

- 프런트 포인터 : 가장 먼저 삽입된 자료의 주소, 삭제 할 때 사용

- 리어 포인터 : 가장 나중에 삽입된 자료의 주소, 삽입 할 때 사용

3) 운영체제의 작업 스케줄링에 사용

7. 그래프(Graph)

1) 정점 V와 간선 E의 두 집합으로 이루어짐

2) 트리는 사이클이 없는 그래프이다.

3) 통신망, 교통망 등에 이용

4) 무방향 그래프 : n(n-1)/2, 방향 그래프 : n(n-1)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

37주차 - 트리

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 트리의 개요

1) 정점(Node)과 선분(Branch)를 이용해 사이클을 이루지 않도록 구성한 그래프

2) 하나의 기억 공간을 노드, 노드와 노드를 연결하는 선을 링크라고 함

3) 족보, 조직도 표현하기 좋음

4) 트리의 용어

- 노드(Node)

- 트리의 기본 요소 자료 항목

- 근 노드(Root Node)

- 맨 위 노드

- 디그리(Degree, 차수)

- 노드에서 뻗어 나온 가지 수

- 트리의 디그리 : 디그리 중 가장 많은 수

- 단말 노드(Terminal Node) = 잎 노드(Leaf Node)

- 자식이 하나도 없는 노드(Degree = 0)

- 자식 노드(Son Node)

- 부모 노드(Parent Node)

- 형제 노드(Brother Node, Sibling)

- 동일한 부모를 갖는 노드

- 깊이(Depth)

- 트리의 깊이

2. 운행법(Traversal)

1) 루트의 위치에 따라 3가지로 나뉨

- Preorder : 위 왼 오

- Inorder : 왼 위 오

- Postorder : 왼 오 위

3. 수식의 표기법

1) 루트의 위치에 따라 3가지로 나뉨

- 전위 표기법(PreFix) : 위 왼 오 +AB

- 중위 표기법(InFix) : 왼 위 오 A+B

- 후위 표기법(PostFix) : 왼 오 위 AB+

ex) inorder로 작성된 식 변환

X = A / B \* (C + D) + E

1) prefix로 변환

- 우선순위에 따라 괄호로 묶기

[X = {((A / B) \* (C + D)) + E}]

- 연산자를 왼쪽으로 옮김

=[X +{\*(/(A B) +(C D)) E}]

- 괄호 삭제

=X+\*/AB+CDE

2) postfix로 변환

- 우선순위에 따라 괄호로 묶기

[X = {((A / B) \* (C + D)) + E}]

- 연산자를 오른쪽으로 옮김

[X {((A B)/ (C D)+)\* E}+]=

- 괄호 삭제

XAB/CD+\*E+=

ex) postfix로 작성된 식 변환

A B C - / D E F + \* +

- 인접한 피연산자 두개와 오른쪽 연산자를 괄호로 묶음

[{A (B C -) /} {D (E F +) \*} +]

- 연산자를 가운데로 옮김

[{A / (B - C)} + {D \* (E + F)}]

- 필요없는 괄호 삭제

A / (B - C) + D \* (E + F)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

38주차 - 정렬(Sort)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 삽입 정렬(Insertion Sort)

1) 이미 순서화된 파일에 새로운 하나의 레코드를 순서에 맞게 삽입시켜 정렬

2) 1회전 : 두번쨰 값을 앞의 값들과 첫번째 값부터 비교

2회전 : 세번째 값을 앞의 값들과 첫번째 값부터 비교

3회전 : 네번째 값을 앞의 값들과 첫번째 값부터 비교

3) 앞의 값을 비교하고 첫번째에 들어가면 한칸씩 뒤로 밀림

ex) 8 5 6 2 4

1회전 : 5 8 6 2 4

2회전 : 5 6 8 2 4

3회전 : 2 5 6 8 4

4회전 : 2 4 5 6 8

4) 평균, 최악 : O(n^2)

2. 쉘 정렬(Shell Sort)

1) 삽입 정렬을 확장한 개념

2) 임의의 값 h 값으로 서브파일 구성 후 각 서브파일에서 삽입 정렬 후 과정 반복

3) 입력 파일이 부분적으로 정렬 되어있는 경우 유리

4) 평균 : O(n^1.5) 최악 : O(n^2)

3. 선택 정렬(Selection Sort)

1) 맨 앞 값을 뒤에 값과 계속 비교하여 최소값을 앞으로 가져다 놓음

2) 1회전 끝날때마다 맨 앞이 최소값이 됨

ex) 8 5 6 2 4

1회전 : 8 5 6 2 4 -> 5 8 6 2 4 -> 2 8 6 5 4 (2는 정렬됨)

2회전 : 2 8 6 5 4 -> 2 6 8 5 4 -> 2 5 8 6 4 -> 2 4 8 6 5 (2,4는 정렬됨)

3회전 : 2 4 8 6 5 -> 2 4 6 8 5 -> 2 4 5 8 6 (2,4,5는 정렬됨)

4회전 : 2 4 5 8 6 -> 2 4 5 6 8

3) 평균, 최악 : O(n^2)

4. 버블 정렬(Bubble Sort)

1) 인접한 두 개를 비교하여 위치 교환

2) 계속 정렬 여부를 플래그 비트로 결정

3) 끝 값부터 정렬됨

ex) 8 5 6 2 4

1회전 : 8 5 6 2 4 -> 5 8 6 2 4 -> 5 6 8 2 4 -> 5 6 2 8 4 -> 5 6 2 4 8 (8은 정렬됨)

2회전 : 5 6 2 4 8 -> 5 2 6 4 8 -> 5 2 4 6 8 (6,8은 정렬됨)

3회전 : 5 2 4 6 8 -> 2 5 4 6 8 -> 2 4 5 6 8

4회전 : 2 4 5 8 6

4) 평균, 최악 : O(n^2)

5. 퀵 정렬(Quick Sort)

1) 많은 자료 이동을 없애고 하나의 파일을 부분적으로 나누어 가며 정렬하는 방법

2) 키를 기준으로 작은 값은 왼쪽, 큰 값은 오른쪽 서브파일로 분해

3) 가장 빠른 방식, 스택이 필요

4) 분할과 정복을 이용

- 분할 : 기준값이 피봇을 중심으로 2개 부분집합으로 나눔

- 정복 : 피봇보다 작은 원소들은 왼쪽 피봇보다 큰 원소들은 오른쪽으로 정렬

5) 크기가 나눠질 수 없을 때까지 분할 정복 수행

6) 평균 : O(nlog2n), 최악 : O(n^2)

6. 힙 정렬(Heap Sort)

1) 전이진트리(완전이진트리)를 이용한 정렬

2) 전이진트리를 힙트리로 변환하여 정렬

3) 평균, 최악 : O(nlog2n)

4) 역순으로 부모와 비교 후 교환을 계속 반복

7. 2-Way 합병 정렬(Merge Sort)

1) 이미 정렬되어 있는 두개의 파일을 한개의 파일로 합병

2) 2개씩 묶고 각 쌍에 대하여 정렬 후 합치고 다시 정렬 반복

3) 평균, 최악 : O(nlog2n)

ex) 71 2 38 5 7 61 11 26 53 42

1회전 : (71 2) (38 5) (7 61) (11 26) (53 42) -> (2 71) (5 38) (7 61) (11 26) (42 53)

2회전 : (2 71 5 38) (7 61 11 26) (42 53) -> (2 5 38 71) (7 11 26 61) (42 53)

3회전 : (2 5 38 71 7 11 26 61) (42 53) -> (2 5 7 11 26 38 61 71) (42 53)

4회전 : (2 5 7 11 26 38 61 71 42 53) -> (2 5 7 11 26 38 42 53 61 71)

8. 기수 정렬(Radix Sort) = Bucket Sort

1) 큐를 이용하여 자릿수별로 정렬하는 방식

2) 키값을 분석하여 같은 수 또는 문자끼리 순서에 맞는 버킷에 분배하여 버킷 순서대로 꺼내어 정렬

3) 평균, 최악 : O(dn)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

39주차 - 검색-이분검색/해싱

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 이분 검색(Binary Search)

1) 전체 파일을 두 개의 서브파일로 분리해가면서 검색

2) 반드시 순서화된 파일

3) key 값을 중간 레코드 key값과 비교

4) 반복할 때마다 검색 대상 데이터가 반으로 줄어듦

5) 중간 레코드 번호 M = (F+L)/2 (F : 첫 번째 레코드 번호, L : 마지막 레코드 번호)

- M : 정수부분만 취한다.(반올림 X)

ex) 1~100 중 15찾는데 걸리는 횟수? (F = 1, L = 100)

- M = (1+100)/2 = 50.5 = 50

1회 비교 : 15가 50보다 작음 (F = 1, L = 49)

- M = (1+49)/2 = 25

2회 비교 : 15가 25보다 작음 (F = 1, L = 24)

- M = (1+24)/2 = 12.5 = 12

3회 비교 : 15가 12보다 큼 (F = 13, L = 24)

- M = (13+24)/2 = 18.5 = 18

4회 비교 : 15가 18보다 작음 (F = 13, L = 17)

- M = (13+17)/2 = 15

5회 비교 : 15를 찾음

5회 비교해야 됨

2. 해싱

1) 해시 테이블이라는 기억 공간을 할당하고 해시함수를 이용해 키에 대한 해시 테이블 내의 홈 주소를 계산 후

저장 또는 검색을 한다.

2) 해시 테이블

- 레코드를 한 개 이상 보관할 수 있는 버킷들로 구성된 기억공간

- 보조기억 장치 또는 주기억장치에 구성 가능

- 버킷 : 하나의 주소를 갖는 파일의 한 구역, 버킷의 크기는 같은 주소에 포함될 수 있는 레코드 수를 의미함

- 슬롯 : 한 개의 레코드를 저장할 수 있는 공간, 슬롯이 모여 버킷을 형성함

- 충돌 현상 : 서로 다른 두 개 이상의 레코드가 같은 주소를 같는 현상

- Synonym : 충돌로 인해 같은 홈 주소를 같는 레코드의 집합

- Overflow : 홈 주소 버킷 내에 저장할 기억공간이 없는 상태

3) 해싱 함수

- 제산법(Division)

- 키를 해시표의 크기보다 큰 수 중 가장 작은 소수로 나눈 나머지를 홈주소로 함, h(k) = K mod Q

- 제곱법(Mid-Square)

- 키 값을 제곱 후 중간값을 홈 주소로 함

- 폴딩법(Folding)

- 키 값을 여러 부분으로 나눈 후 각 값을 더하거나 XOR한 값을 홈 주소로 함

- 기수 변환법(Radix)

- 진수를 다른 진수로 변환시켜 주소 크기를 초과한 높은 자릿수를 절단, 주소 범위에 맞게 변환

- 대수적 코딩법(Algebric Coding)

- 각 자리 비트를 다항식의 계수로 간주하고 정의된 다항식으로 나누어 얻은 나머지 다항식의 계수를 홈주소로 함

- 숫자 분석법(Digit Analysis, 계수 분석법)

- 키 값을 이루는 숫자의 분포를 분석해 고른 자리를 필요한 만큼 골라서 홈주소로 함

- 무작위법(Random)

- 난수를 발생시켜 나온 값을 홈 주소로 함

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

40주차 - 데이터베이스 개요

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 데이터 저장소

1) 데이터를 논리적인 구조로 조직화하거나 물리적 공간에 구축한 것

2) 논리 데이터저장소, 물리 데이터저장소로 나뉨

- 논리 데이터저장소 : 데이터, 연관성, 제약조건 식별하여 논리적인 구조로 조직화

- 물리 데이터저장소 : 논리 데이터저장소를 환경의 물리적 특성을 고려하여 하드웨어적인 저장장치에 저장

2. 데이터베이스

1) 통합된 데이터(Integrated Data) : 자료의 중복을 배제한 데이터 모임

2) 저장된 데이터(Stored Data) : 컴퓨터가 접근할 수 있는 저장 매체에 저장된 자료

3) 운영 데이터(Operational Data) : 조직 고유 업무 수행하는 데 존재 가치가 확실하고 없어선 안될 반드시 필요한 자료

4) 공용 데이터(Shared Data) : 여러 응용 시스템이 공동으로 소유,유지 자료

3. DBMS

1) 데이터베이스 관리 시스템

2) 데이터의 종속성과 중복성 문제를 해결하기 위해 제안된 시스템

3) 모든 응용 프로그램들이 데이터베이스를 공용할 수 있도록 관리

4) 필수기능 3가지 정의, 조작, 제어

- 정의(Definition)

- 데이터 타입과 구조에 대한 정의, 이용 방식, 제약 조건 명시

- 조작(Manipulation)

- 입력,갱신,삭제 등 사용자와 데이터베이스 사이의 인터페이스 수단 제공

- 제어(Control)

- 데이터 무결성이 유지되도록 제어

- 보안, 권한, 병행 제어

4. DBMS 장단점

1) 장점

- 데이터의 논리적,물리적 독립성 보장

- 논리적 독립성 : 응용 프로그램과 데이터베이스 독립, 데이터 논리적 구조를 변경해도 응용 프로그램은 변하지 않음

- 물리적 독립성 : 응용 프로그램과 보조기억장치 독립, 데이터베이스 업그레이드 해도 응용 프로그램에 영향 주지 않음

- 중복 피해서 기억 공간 절약

- 자료 공동 이용

- 데이터 일관성, 무결성, 보안 유지

- 데이터 표준화, 통합, 최신화, 실시간 처리 관리

2) 단점

- 데이터베이스 전문가가 부족함

- 전산화 비용 증가

- 파일 백업과 회복이 어렵고 복잡

5. 스키마

1) 데이터베이스 구조와 제약 조건에 관한 명세를 기술한 메타데이터

2) 사용자 관점에 따라 외부 스키마, 개념 스키마, 물리 스키마로 나뉨

- 외부 스키마

- 각 개인의 입장에서 필요로 하는 데이터베이스의 논리적 구조 정의

- 개념 스키마

- 전체적인 논리적 구조 모든 응요 프로그램이나 사용자가 필요로 함

- 단 하나만 존재

- 물리 스키마

- 물리적 저장장치의 입장에서 본 데이터베이스 구조가

- 레코드 형식 정의, 순서 정의

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

42주차 - 절차형 SQL

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 절차형 SQL의 개요

1) C, JAVA 등 프로그래밍 언어와 같이 연속적인 실행이나 분기, 반복 등 제어가 가능한 SQL

2) 일반적인 프로그래밍 언어에 비해 효율은 떨어지나 단일 SQL 문장으로 처리하기 어려운 연속적인 작업 처리

3) DBMS 엔진에서 직접 실행되서 입 출력 패킷이 적음

4) BEGIN ~ END 형식의 블록 구조, 따라서 기능별 모듈화 가능

5) 절차형 SQL 종류

- 프로시저 : 특정 기능을 수행하는 트랜잭션 언어, 호출을 통해 미리 저장된 SQL 작업 수행

- 트리거 : 이벤트가 발생할 때마다 관련 작업 자동 수행

- 사용자 정의 함수 : 종료 시 예약어 Return을 사용해 처리 결과를 단일값으로 반환

2. 절차형 SQL의 테스트와 디버깅

1) 디버깅을 통해 기능의 적합성 여부 검증, 실행을 통해 결과를 확인하는 테스트 수행

2) 테스트 전에 구문오류나 참조오류 존재 여부 확인

3) SHOW 명령어를 통해 상세 오류 내용 확인

4) 테스트를 통해 오류 발견 후 디버깅을 통해 오류 추적

3. 쿼리 성능 최적화

1) SQL코드를 최적화하는 것

2) 성능측정도구인 APM을 사용해 최적화 할 쿼리 선정

3) 최적화 할 쿼리에 대해 옵티마이저가 수립한 실행 계획 검토, SQL코드와 인덱스 재구성

- 옵티마이저 : 최적의 경로를 찾아 주는 모듈

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

44주차 - 단위 모듈 테스트

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 단위 모듈 테스트의 개요

1) 모듈이 정해진 기능을 정확히 수행하는지 검증

2) 단위 테스트라고도 하고, 화이트박스 테스트와 블랙박스 테스트 기법 사용

- 화이트박스 테스트 : 모듈의 소스코드를 오픈시킨 상태에서 모든 논리적인 경로를 테스트

- 블랙박스 테스트 : 특정 기능이 완전히 작동되는 것을 입증하는 테스트

3) 시스템 수준의 오류는 잡아낼 수 없다.

2. 테스트 케이스

1) 사용자의 요구사항을 정확하게 준수했는지 확인하기 위한 테스트 항목에 대한 명세서

2) 필요한 입력 데이터, 테스트 조건, 예상 경로 등

3) ISO/IEC/IEEE 29119-3 표준에 따른 테스트 케이스의 구성요소(식테 입출 환경 특의) 식사데이트, 입출력 환경이 특의하네

- 식별자 : 항목 식별자, 일련번호

- 테스트 항목 : 테스트 대상

- 입력 명세 : 입력 데이터

- 출력 명세 : 예상되는 출력 결과

- 환경 설정 : 필요한 하드웨어나 소프트웨어 환경

- 특수 절차 요구 : 테스트 케이스 수행 시 특별히 요구되는 절차

- 의존성 기술 : 테스트 케이스 간의 의존성

4. 테스트 프로세스

1) 모든 작업이 테스트의 목적과 조건을 달성할 수 있도록 하는 과정

2) 테스트 프로세스 5단계

- 계획 및 제어 단계 : 계획 수립

- 분석 및 설계 단계 : 목표 구체화, 테스트 시나리오(테스트 케이스 묶음)와 테스트 케이스 작성

- 구현 및 실현 단계 : 테스트 케이스를 조합하여 테스트 프로시저(테스트 케이스 실행순서)에 명세

- 평가 단계 : 평가

- 완료 단계 : 결과 기록 및 저장

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

45주차 - 개발 지원 도구

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 통합 개발 환경(IDE)

1) 코딩, 디버그, 컴파일, 배포 등 개발과 관련된 모든 작업을 하나의 프로그램에서 처리하는 소프트웨어적인 개발 환경

2)프로그램 - 개발사 - 플랫폼 - 운영체제 - 지원언어

-------------------------------------------------------------------------------------

이클립스 - Eclipse Foundation - 크로스 플랫폼 - Windows - Java,C,C++

IBM Linux PHP,JSP 등

MacOS 등

-------------------------------------------------------------------------------------

비주얼 - Microsoft - Win32,Win64 - Windows - Basic,C,C++

스튜디오 C#, .NET 등

-------------------------------------------------------------------------------------

엑스코드 - Apple - Mac,iPhone - MacOS - C,C++

iOS C#,Java

AppleScript 등

-------------------------------------------------------------------------------------

안드로이드 - Google - Android - Windows - Java,C,C++ 등

스튜디오 Linux

MacOS

-------------------------------------------------------------------------------------

IDEA - JetBrains - 크로스 플랫폼 - Windows - Java,JSP

Linux XML,Go,Kotlin

MacOS PHP 등

3) 대표적인 기능

- 코딩 : C, JAVA 등 프로그래밍 언어로 프로그램 작성

- 컴파일 : 고급언어를 컴퓨터에서 실행 가능한 형태로 변환

- 디버깅 : 오류나 잘못된 동작, 버그를 찾아 수정

- 배포 : 소프트웨어를 사용자에게 전달

2. 빌드 도구

1) 소스 코드 파일들을 컴퓨터에서 실행할 수 있는 제품 소프트웨어로 변환하는 과정 또는 결과물

- 소스 코드 파일을 실제 실행할 수 있는 파일로 변환해줌

2) 빌드 도구는 전처리(컴파일에 앞서 주석제거하거나 매크로 처리하는 과정), 컴파일을 수행하는 소프트웨어

- Ant

- 아파치에서 개발, 자바 프로젝트의 공식적인 빌드 도구

- XML 기반의 빌드 스크립트 사용, 자유도와 유연성이 높아 복잡한 빌드 환경에서도 대처 가능

- 정해진 규칙, 표준이 없음, 스크립트 재사용 어려움

- Maven

- 아파치에서 개발, Ant의 대안

- 규칙, 표준이 존재 예외 사항만 기록하면 됨, 컴파일과 빌드 동시 수행

- 의존성을 설정해 라이브러리 관리

- Gradle

- Ant와 Maven을 보완

- 한스 도커 외 6인 개발자가 공동 개발

- 안드로이드 스튜디오 공식 빌드 도구

- 의존성 활용, 그루비 기반 빌드 스크립트 사용

3. 기타 협업 도구

1) 개발에 참여하는 사람들이 서로 다른 작업 환경에서 프로젝트 수행하도록 도와주는 툴

2) 협업 소프트웨어, 그룹웨어

3) 종류

- 프로젝트 및 일정 관리

- 구글 캘린더, 분더리스트, 트렐로, 지라, 플로우

- 정보 공유 및 커뮤니케이션

- 슬랙, 잔디, 태스크월드

- 디자인

- 스케치, 제플린

- 기타

- 에버노트 : 아이디어 공유

- 스웨거 : API 문서화하여 개발자들 간 협업 도움

- 깃허브 : 깃의 웹호스팅 서비스

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

46주차 - 소프트웨어 패키징

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 소프트웨어 패키징의 필요

1) 모듈별로 생성한 실행 파일을 묶어 배포용 설치 파일을 만드는 것

2) 사용자 중심

3) 모듈화(기능별로 나누어)하여 패키징

2. 패키징 시 고려사항

1) 최소환경 정의

2) Managed Service(유지보수) 형태로 제공

3) 내부 콘텐츠에 대한 암호화 및 보안, 적합한 암호화알고리즘

4) DRM 연동 고려사항

3. 패키징 작업 순서

1) 주기 별로 패키징한 결과물은 테스트 서버에 배포

2) 최종 패키징한 결과물은 고객이 사용할 수 있게 온라인, 오프라인으로 배포

- 온라인 : 별도로 마련한 매뉴얼과 함께 고객이 직접 다운받아 사용

- 오프라인 : CD-ROM이나 DVD, USB 등에 매뉴얼과 함꼐 배포

3) 순서(기능 모빌 사이코패스 패배)

- 기능 식별 : 코드 기능 확인

- 모듈화 : 기능 단위로 코드 분류

- 빌드 진행 : 모듈 단위별로 실행 파일 만듦

- 사용자 환경 분석 : 최소 운영 환경 정의

- 패키징 및 적용 시험 : 배포용 파일 형식으로 패키징, 테스트 후 불편사항을 사용자 입장에서 확인

- 패키징 변경 개선 : 확인된 불편 사항을 반영 개선

- 배포 : 배포 시 오류가 발생하면 개발자에게 전달하고 수정 요청

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

48주차 - 디지털 저작권 관리(DRM)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 저작권의 개요

1) 저작물에 대해 창작자가 가지는 배타적 독점적 권리

2. 디지털 저작권 관리(DRM)

1) 원본 컨텐츠가 아날로그인 경우 디지털로 변환 후 패키저에 의해 DRM 패키징 수행

2) 콘텐츠 크기가 작으면 요청할 때 실시간으로 패키징 수행, 크기가 크면 미리 패키징 수행 후 배포

3) 패키징을 수행하면 암호화된 저작권자의 전자서명이 포함됨, 라이선스 정보가 클리어링 하우스에 등록됨

4) 사용자가 사용하기 위해서는 클리어링 하우스에 등록된 라이선스 정보를 통해 사용자 인증과 콘텐츠 사용 권한 소유 여부

확인받아야 함

3. 디지털 저작권 관리의 흐름 및 구성 요소

1) 클리어링 하우스 : 저작권에 대한 사용 권한, 라이선스 발급, 키 관리 등 수행

2) 콘텐츠 제공자 : 저작권자

3) 패키저 : 배포 가능한 형태로 묶어 암호화하는 프로그램

4) 콘텐츠 분배자 : 유통하는 곳이나 사람

5) 콘텐츠 소비자 : 콘텐츠 구매하는 사람

6) DRM 컨트롤러 : 배포된 컨텐츠의 이용 권한 통제하는 프로그램

7) 보안 컨테이너 : 원본을 안전하게 유통하기 위한 전자적 보안 장치

콘텐츠 소비자가 콘텐츠 분배자에게 라이선스 요청하면 클리어링 하우스에 요청하여 클리어링 하우스가

소비자에게 라이선스 발급 해줌

4. 디지털 저작권 관리의 기술 요소(암키암 저 식인 정크랙)

1) 암호화(Encryption)

2) 키관리

3) 암호화 파일 생성(Packager)

4) 식별 기술(Identification)

5) 저작권 표현(Right Expression)

6) 정책 관리

7) 크랙 방지(Tamper Resistance)

8) 인증(Authentication)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

49주차 - 소프트웨어 설치 매뉴얼 작성

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 소프트웨어 설치 매뉴얼의 개요

1) 설치하는 과정에 필요한 내용을 기록한 설명서 안내서

2) 사용자 기준으로 전 과정을 빠짐없이 순서대로 설명

3) 오류는 따로 분류하여 설명

2. 서문

1) 문서이력

- 변경 내용 계속 기록

2) 설치 매뉴얼의 주석

- 주의 사항 참고 사항 기술

- 주의 사항 : 사용자가 반드시 알아야 하는 것

- 참고 사항 : 영향을 미치는 사용자 환경이나 상황에 대해 기술

3) 설치 도구의 구성

- exe, dll, ini, chm 등 설치 관련 파일에 대해 설명

- 설치 과정 및 결과가 기록되는 log폴더에 대해 설명

4) 설치 환경 체크 항목

- 사용자 환경 : CPU, Memory, OS

- 응용 프로그램 : 설치 전 다른 응용 프로그램 종료

- 업그레이드 버전 : 업그레이드 이전 버전 존재 유무 확인

- 백업 폴더 확인 : 데이터 저장 폴더를 확인하여 설치 시 폴더를 동기화시킴

3. 기본사항

1) 소프트웨어 개요 : 주요 기능, UI 그림으로 설명

2) 설치 관련 파일 : 설치에 필요한 파일 설명, exe,ini 파일 설명

3) 설치 아이콘 : 설치 아이콘 설명

4) 프로그램 삭제 : 삭제 방법 설명

5) 관련 추가 정보 : 관련 설치 프로그램, 제작사 등 추가 정보 기술

4. 설치 매뉴얼 작성 방법

1) 누락없이 쉽게 캡처하면서 순서대로 상세히 설명

2) 설치 화명 및 UI

- 설치 실행과 메인 화면 안내창에 대한 내용 기술

3) 설치 이상 메세지 설명

4) 설치 완료 및 결과

5) FAQ

6) 설치 시 점검 사항

7) Network 환경 및 보안

8) 고객 지원 방법

9) 준수정보&제한보증

- Serial 보존(숫자 문자 혼합된 일련번호)

5. 설치 매뉴얼 작성 순서

1) 기능 식별

2) UI 분류

3) 설치 파일/백업 파일 확인

4) Uninstall 절차 확인

5) 이상 Case 확인

6) 최종 매뉴얼 적용

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

50주차 - 소프트웨어 사용자 매뉴얼 작성

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 소프트웨어 사용자 매뉴얼의 개요

1) 사용자가 소프트웨어 사용하는 과정에서 필요한 내용 기술한 안내서

2) 개별적으로 동작이 가능한 컴포넌트 단위로 매뉴얼 작성

3) 컴포넌트 명세서, 컴포넌트 구현 설계서를 토대로 작성

2. 서문

1) 문서 이력

2) 사용자 매뉴얼의 주석

- 주의 사항 참고 사항 기술

- 주의 사항 : 사용자가 반드시 알아야 하는 것

- 참고 사항 : 영향을 미치는 사용자 환경이나 상황에 대해 기술

3) 기록 보관 내용

- 기술 지원이나 추가 정보 얻기 위한 소프트웨어 등록 정보 기술

3. 기본 사항

1) 소프트웨어 개요 : 주요 기능, UI 그림으로 설명

2) 소프트웨어 사용 환경 : 최소환경, PC사양, 최초 구동에 대한 설명

3) 소프트웨어 관리 : 사용 종료 및 관리

4) 모델,버전별 특징 : 별로 UI 및 기능 차이 간략히 요약

5) 기능,인터페이스의 특징 : 기능, 인터페이스 특징 간략히 요약

6) 소프트웨어 구동 환경 : 개발에 사용된 언어, 호환 가능한 운영체제 설명, 설치 후 구동까지 과정을 OS별로 설명

4. 사용자 매뉴얼 작성 방법

1) 누락 없이 캡처

2) 사용자 화면 및 UI

- 주의사항 참고사항 기술

3) 주요 기능 분류

4) 응용 프로그램 및 설정

5) 장치 연동

6) Network 환경

7) Profile 안내

- Profile : 소프트웨어의 구동 환경을 점검하는 파일

- 사용자가 프로필 경로를 변경하거나 위치를 이동하지 않도록 안내

8) 고객 지원 방법

9) 준수정보&제한보증

5. 사용자 매뉴얼 작성 순서

1) 작성 지침 정의

2) 사용자 매뉴얼 구성 요소 정의

3) 구성 요소별 내용 작성

4) 사용자 매뉴얼 검토

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

51주차 - 소프트웨어 버전 등록

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 소프트웨어 패키징의 형상 관리

1) 형상관리(SCM) 변경 사항을 관리하기 위한 활동

- 형상 : 개발 각 과정에서 나오는 프로그램, 데이터, 문서

2) 소프트웨어 개발의 전 단계에 적용, 유지보수 단계에서도 함

3) 여러 방해 요인 최소화 보증

4) 가시성과 추적성을 보장함

5) Git, CVS, Subversion

2. 형상 관리의 중요성

1) 변경 사항 추적,통제 가능

2) 무절제한 변경 방지

3) 여러 개발자 동시에 개발

3. 형상관리 기능 (식 제통 감기)

1) 형상 식별 : 이름,번호 부여하고 트리구조로 구분하여 추적 수정 쉽게 만듦

2) 버전 제어 : 업그레이드 유지보수를 통해 다른 버전 형상 항목 관리, 이를 위해 특정 절차와 도구를 결합

3) 형상 통제 : 변경 요구를 검토하여 현재 기준선 잘 반영되도록 하는것

4) 형상 감사 : 기준선의 무결성 평가하기 위해 확인, 검증을 통해 공식적으로 승인하는 작업

5) 형상 기록 : 기록하고 보고서 작성

4. 소프트웨어 버전 등록 관련 주요 기능(동체가 커저)

1) 저장소(Repository)

2) 가져오기(Import)

- 버전 관리가 되지 않은 아무것도 없는 저장소에 처음으로 파일 복사

3) 체크아웃

- 프로그램 수정위해 저장소에서 파일을 받아옴

4) 체크인

- 수정 완료 후 저장소 파일을 새로운 버전으로 갱신

5) 커밋

- 체크인 수행할 때 이전에 갱신된 내용이 있는 경우 충돌 알리고 diff도구를 이용해 수정 후 갱신

6) 동기화(Update)

- 저장소에 있는 최신 버전으로 자신의 작업 공간을 동기화한다.

5. 소프트웨어 버전 등록 과정

1) 가져오기(Import)

2) 인출(Check-Out)

3) 예치(Commite)

4) 동기화(Update)

5) 차이(Diff)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

52주차 - 소프트웨어 버전 관리 도구

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 공유 폴더 방식

1) 버전 관리 자료가 로컬 컴퓨터의 공유 폴더에 저장됨

2) 파일의 변경사항을 데이터베이스에 기록하여 관리

3) 종류

- SCCS, RCS(시간에 따른 파일 변화 과정 관리,동시수정방지), PVCS, QVCS (PV,QV,R,SC) pqrs

2. 클라이언트/서버 방식

1) 버전 관리 자료가 중앙 시스템(서버)에 저장되어 관리되는 방식

2) 서버의 자료를 개발자별로 자신의 pc에 복사 후 변경된 내용을 서버에 반영

3) 모든 버전 관리는 서버에서 수행

4) 서버에 문제 생기면 작업 중단

5) 종류

- CVS, SVN(Subversion), CVSNT, Clear Case, CMVC, Perforce

3. 분산 저장소 방식

1) 버전 관리 자료가 하나의 원격 저장소와 분산된 로컬저장소에 함께 저장되어 관리

2) 로컬 저장소에 복사, 작업 후 로컬 저장소에 우선 반영한 다음 원격 저장소에 반영

3) 원격 저장소에 문제가 생겨도 로컬 저장소를 이용할 수 있다.

4) 종류

- Git, GNU arch, DCVS, Bazaar, Mercurial, TeamWare, Bitkeeper, Plastic SCM

4. Subversion(서브버전,SVN)

1) CVS 개선 방법, 아파치

2) 서버에는 최신 버전 파일과 변경 내역 관리

3) 모든 개발은 trunk 디렉터리에서 수행, 추가 작업은 branches 디렉터리 안에 별도 디렉터리 만들어 완료 후 trunk와 병합

4) 커밋할때마다 리비전 1 증가

5) 오픈소스

6) 주요 명령어

- add : 새로 버전 관리 대상 등록

- commit : 클라이언트 소스파일을 서버 소스파일에 적용

- update : 서버의 최신 commit 이력을 클라이언트 소스 파일에 적용

- checkout : 버전 관리 정보와 소스파일을 서버에서 클라이언트로 받아온다.

- lock/unlock : 서버의 파일이나 디렉토리를 잠그거나 해제

- import : 아무것도 없는 서버 저장소에 소스파일 저장, 한번만 사용

- export : 버전관리정보를 제외한 순수한 소스파일 서버에서 받아옴

- info : 지정한 파일에 대한 위치,마지막 수정일자 등 정보 표시

- diff : 지정된 파일이나 경로에 대해 이전 리비전과의 차이 표시

- merge : 다른 디렉터리에서 작업된 버전 관리 내역을 기본 개발 작업과 병합

5. Git(깃)

1) 리누스 토발즈-리눅스 커널 개발, 주니오 하마노-유지보수

2) 로컬 저장소, 원격 저장소

- 로컬 저장소 : 실제 개발하는 장소, 버전 관리 수행

- 원격 저장소 : 협업을 위해 버전 공동 관리하는 곳

3) 브랜치를 이용해 틀에 영향을 주지 않으면서 다양한 기능 테스팅 가능

- 브랜치 : 새로운 브랜치에서 작업 수행 후 마스터 브랜치에 병합, 독립적인 작업이 가능

4) 파일의 변화를 스냅샷으로 저장, 스냅샷은 이전 스냅샷의 포인터를 가짐

- 스냅샷 : 영문자,숫자가 혼합된 40자리 문자열

5) 주요 명령어

- add : 작업 내역을 지역 저장소에 저장하기 위해 스테이징 영역에 추가, '--all' 옵션으로 모든 파일 가능

- commit : 작업 내역을 지역 저장소에 저장

- branch : 새 브랜치 생성, 최초 commit 시 마스터 브랜치 생성, '-d' 옵션으로 삭제 가능

commit 할 때마다 해당 브랜치는 가장 최근 commit 내용을 가리킴

- checkout : 지정한 브랜치로 이동, 현재 작업 브랜치는 head포인터가 가리킴,

checkout을 통해 head포인터를 지정한 브랜치로 이동

- merge : 지정한 브랜치의 변경 내역을 현재 head포인터가 가리키는 브랜치에 반영

- init : 지역 저장소 생성

- remote add : 원격 저장소에 연결

- push : 로컬저장소 내용을 원격 저장소에 반영

- fetch/clone : 원격저장소 변경 이력만을 지역 저장소로 가져와 반영

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

53주차 - 빌드 자동화 도구

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 빌드 자동화 도구의 개념

1) 소스 코드 파일들을 컴파일한 후 여러 개의 모듈을 묶어 실행 파일로 만드는 과정을 포함하여 테스트 및 배포 자동화

2) 지속적인 통합 개발 환경에서 유용

3) 종류 Ant, Make, Maven, Gradle, Jenkins

2. Jenkins

1) JAVA 기반 오픈소스

2) 서블릿 컨테이너에서 실행되는 서버 기반 도구

- 서블릿 컨테이너 : 클라이언트 요청 처리 위해 서버측에서 실행되는 작은 프로그램 실행 및 서블릿 생명주기 관리

3) SVN, Git 등 형상 관리 도구 연동 가능

4) 여러 컴퓨터를 이용한 분산 빌드나 테스트 가능

3. Gradle

1) Groovy 기반 오픈 소스, 안드로이드 앱 개발 환경에서 사용

2) 플러그인 설정하면, JAVA,C++ 등의 언어도 빌드 가능

3) DSL을 스크립트 언어로 사용

4) 태스크 단위로 실행

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

54주차 - 애플리케이션 테스트

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 애플리케이션 테스트의 개념

1) 애플리케이션 잠재 결함 찾아내는 행위

2) 앱이 요구사항을 만족하는지 확인 하고 기능을 정확히 수행하는지 검증한다.

확인 : 사용자 입장

검증 : 개발자 입장

2. 애플리케이션 테스트의 기본 원리

1) 결합이 없다고 증명할 수는 없다. 완벽한 테스트 불가능

2) 파레토 법칙 : 20%의 코드에서 전체 80%의 결함이 발견, 결함 집중

3) 살충제 패러독스(약에 내성이 생기는 것)을 방지하기 위해 테스트를 지속적으로 보완 및 개선

4) 오류-부재의 궤변 : 결함이 없어도 요구사항을 만족못하면 품질이 높다고 말할 수 없음

5) 테스트는 작은 부분에서 시작하여 점점 확대

6) 테스트는 개발자와 관계없는 별도 팀에서 진행

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

55주차 - 애플리케이션 테스트의 분류

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 프로그램 실행 여부에 따른 테스트

1) 프로그램 실행 여부에 따라 정적 테스트와 동적 테스트로 나뉨

- 정적 테스트

- 실행하지 않고 명세서나 소스코드 대상 테스트

- 개발 초기에 결함 발견

- 워크스루,인스펙션,코드 검사

- 동적 테스트

- 실행하여 오류 찾는 테스트

- 개발 모든 단계에서 가능

- 블랙박스 테스트, 화이트박스 테스트

2. 테스트 기반에 따른 테스트(명구경이구만!!!)

1) 명세 기반 테스트 - 사용자 요구사항에 대한 명세를 빠짐없이 테스트 케이스로 만들어 구현하는지 확인

2) 구조 기반 테스트 - 내부 논리 흐름에 따라 테스트 케이스 작성하는지 확인

3) 경험 기반 테스트 - 경험 기반하여 명세가 불충분하거나 시간에 제약 있는 경우 효과적

3. 시각에 따른 테스트

1) 검증 테스트 - 개발자 입장, 생산 과정 테스트

2) 확인 테스트 - 사용자 입장, 결과 테스트

4. 목적에 따른 테스트 (강도한테서 안전하게 구조됐는데 회복하다가 성병 걸려서 회귀함)

1) 강도 - 과부하 시 정상적으로 실행되는지

2) 안전 - 불법적 침입으로부터 보호할 수 있는지

3) 구조 - 논리적인 경로 소스코드 복잡도 평가

4) 회복 - 결함 생겨서 실패할 경우 올바르게 복구되는지

5) 성능 - 실시간 성능, 효율성, 응답시간,처리량 평가

6) 병행 - 기존 소프트웨어와 결과 비교

7) 회귀 - 변경 코드에 결함이 없는지

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

56주차 - 테스트 기법에 따른 애플리케이션 테스트

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 화이트박스 테스트

1) 모듈의 원시 코드를 오픈시킨 상태에서 모든 경로 테스트하여 테스트 케이스 설계

2) 설계된 절차에 초점을 둔 구조적 테스트, 테스트 과정의 초기에 적용

3) 모듈 안을 직접 관찰

4) 모든 문장 실행

2. 화이트박스 테스트 종류

1) 기초 경로 검사

- 대표 화이트박스 테스트 기법

- 논리적 복잡성 측정

2) 제어 구조 검사

- 조건 검사 : 모듈 내 논리적 조건 테스트

- 루프 검사 : 반복구조에 초점을 맞춰 테스트

- 데이터 흐름 검사 : 변수 정의와 변수 사용 위치에 초점을 맞춰 테스트

3. 화이트박스 테스트의 검증 기준

1) 문장 검증 기준 : 모든 소스코드가 한번이상 수행

2) 분기 검증 기준 : 결정 검증 기준이라고 불리며, 조건문에 대해 조건이 True,false인 경우 한번이상 수행

3) 조건 검증 기준 : 소스 코드 조건문에 포함된 개별 조건식 결과가 True,false인 경우 한번이상 수행

4) 분기/조건 기준 : 둘 합친거

4. 검증 기준 종류

1) 기능 기반 커버리지 : 실제 테스트 수행된 기능 수/전체 기능 수

2) 라인 커버리지 : 테스트시나리오가 수행한 소스코드라인수/전체소스코드라인수

3) 코드 커버리지 : 소스코드의 구문,분기,조건 등의 구조 코드 자체가 얼마나 테스트 되었는지 측정, =화이트박스 테스트

5. 블랙박스 테스트

1) 특정 기능이 수행되는지 테스트, 기능 테스트라고도 함

2) 구조를 고려하지 않으므로 모듈의 요구나 명세를 기초로 테스트 케이스 작성

3) 인터페이스에서 실시되는 테스트

6. 블랙박스 테스트 종류

1) 동치 분할 검사

- 입력 자료에 초점을 맞춰 테스트 케이스 만듦

- 타당한 자료와 타당하지 않은 자료 입력하고 결과 확인

2) 경계값 분석

- 경계값에서 오류가 발생될 확률이 높음

- 입력 조건의 경계값을 테스트 케이스로 선정

3) 원인-효과 그래프 검사

- 출력에 영향을 미치는 상황 분석하여 효용성 높은 테스트 케이스 선정

4) 오류 예측 검사

- 경험이나 감각으로 테스트

- 보충적 검사 기법, 데이터 확인 검사

5) 비교 검사

- 여러 버전의 프로그램에 동일한 자료 제공하여 동일한 결과가 나오는지 검사

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

57주차 - 개발 단계에 따른 애플리케이션 테스트

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 개발 단계에 따른 애플리케이션 테스트

1) 개발 단계에 따라 단위 테스트, 통합 테스트, 시스템 테스트, 인수 테스트로 나뉨

2) 애플리케이션 테스트는 소프트웨어 개발 단계부터 테스트 수행

3) 요구사항->분석->설계->구현->단위 테스트->통합 테스트->시스템 테스트->인수 테스트

4) V모델

2. 단위 테스트

1) 코딩 직후 모듈이나 컴포넌트에 초점을 맞춰 테스트

2) 인터페이스,외부적I/O,자료구조,오류,경계조건 등 검사

3) 요구사항 기반으로 한 기능성 테스트 최우선

4) 구조기반테스트와 명세기반테스트로 나뉨

- 구조기반테스트 : 프로그램 내부 복잡도 검증하는 화이트박스테스트, 제어 흐름, 조건 결정

- 명세기반테스트 : 목적 및 실행 코드 기반의 블랙박스테스트, 동등 분할, 경계값분석

5) 발견 가능한 오류

- 알고리즘 오류에 따른 원치 않는 결과

- 무한루프

- 틀린 계산 수식에 의한 잘못된 결과

3. 통합 테스트

1) 단위 테스트가 완료된 모듈을 결합하여 하나의 시스템으로 완성시키는 과정 테스트

4. 시스템 테스트

1) 개발된 소프트웨어가 해당 컴퓨터 시스템에서 완벽하게 수행하는가

2) 기능적,비기능적 요구사항 구분하여 각각 만족하는지 테스트

- 기능적 요구사항 : 요구사항 명세서,비지니스 절차,유스케이스 등 명세서 기반 블랙박스 테스트

- 비기능적 요구사항 : 성능테스트,외복,보안 구조적 요소에 대한 화이트박스 테스트

5. 인수 테스트

1) 사용자 요구사항을 충족하는지에 중점을 두고 테스트

2) 사용자가 직접 테스트

3) 종류

- 사용자 인수 테스트 : 사용자가 시스템 사용의 적절성 여부 확인

- 운영상의 인수 테스트 : 시스템 관리자가 수행, 백업복원,점검 등 확인

- 계약 인수 테스트 : 계약상의 인수/검수 조건 준수하는지 여부 확인

- 규정 인수 테스트 : 정부 지침 규정에 맞게 개발됐는지 확인

- 알파 테스트 : 개발자 장소에서 사용자가 개발자 앞에서 행하는 테스트

- 베타 테스트 : 선정된 최초 사용자가 여러 사용자 앞에서 행하는 테스트,필드테스팅,개발자에게 주기적 보고

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

58주차 - 통합 테스트

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 통합 테스트

1) 단위 테스트가 끝난 모듈 통합하는 과정에서 오류 테스트

2) 비점진적,점진적 방법이 있음

- 비점진적 통합 방식

- 단계적 통합 절차 없이 모든 모듈이 결합되어 있는 프로그램 전체 테스트, 빅뱅 통합 테스트 등이 있음

- 빅뱅 통합 테스트 : 단위 테스트 끝난 모듈 한번에 통합, 소규모나 일부만 테스트할 때 사용

- 규모가 작은 소프트웨어에 유리, 단시간 내 테스트 가능, 따라서 오류 발견 등이 어려움

- 점진적 통합 방식

- 모듈 단위로 단계적으로 통합, 상향식,하향식,혼합식이 있음

- 오류 발견 쉽고, 완전 검사할 확률 높음

2. 하향식 통합 테스트

1) 상위->하위 모듈 통합하면서 테스트

2) 깊이 우선 통합법, 넓이 우선 통합법이 있다.

- 깊이 우선 통합법 : 왼쪽깊이 내려가면서 검사하는거

- 넓이 우선 통합법 : 위부터 왼쪽부터 수평으로 검사

3) 테스트 초기부터 사용자에게 시스템 구조 보여줄 수 있음

4) 상위 모듈에서 테스트 케이스 사용 어려움

5) 하향식 순서

- 주요 제어 모듈은 작성된 프로그램 사용, 주요 제어 모듈의 종속 모듈은 스텁으로 대체

- 깊이,넓이 우선 통합방식에 따라 하위 모듈인 스텁들이 한번에 하나씩 실제 모듈로 교체

- 모듈이 통합 될 때마다 테스트

- 새로운 오류가 없음을 보증하기 위해 회귀 테스트 실시

3. 상향식 통합 테스트

1) 하위->상위 모듈 통합하면서 테스트

2) 스텁은 필요하지 않지만 하나의 주요 제어 모듈과 관련된 종속 모듈의 그룹인 클러스터가 필요

3) 상향식 순서

- 하위 모듈을 클러스터로 결합

- 상위 모듈에서 입출력을 확인하기 위해 더미 모듈인 드라이버를 작성

- 통합된 클러스터 단위로 테스트

- 테스트가 완료되면 클러스터는 상위로 이동하여 결합하고 드라이버는 실제 모듈로 대체된다.

4. 드라이버VS스텁

- 둘 다 개발과 테스트 병행할 때 사용

1) 드라이버

- 하위 모듈 호출하는 도구, 매개변수 전달, 테스트 후 결과 도출

- 상위 모듈 없이 하위 모듈이 있는 경우 하위 모듈 구동

- 상향식 테스트

- 이미 존재하는 하위 모듈과 존재하지 않는 상위 모듈 간의 인터페이스 역할

2) 스텁

- 타 모듈의 기능을 단순히 수행하는 도구, 일시적으로 필요한 조건만을 가지고 있는 시험용 모듈

- 싱위 모듈은 있지만 하위 모듈이 없는 경우 하위 모듈 대체

- 하향식 테스트

- 일시적으로 필요한 조건만을 가지고 임시로 제공되는 가짜모듈 역할, 간단하여 작성하기 쉬움

5. 혼합식 통합 테스트

1) 하위 수준에서는 상향식 통합, 상위 수준에서는 하향식 통합을 사용하는 샌드위치식 통합 테스트

6. 회귀 테스팅

1) 이미 테스트 완료한 프로그램의 테스팅을 반복하는 것

2) 새로운 오류가 있는지 검사

3) 변경된 부분만 테스트할 수 있는 테스트 케이스 선정

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

60주차 - 테스트 케이스/테스트 시나리오/테스트 오라클

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 테스트 케이스

1) 사용자의 요구사항을 정확하게 준수했는지 확인하기 위한 테스트 항목에 대한 명세서

2) 시스템 설계 단계에서 작성하는 것이 이상적

2. 테스트 케이스 작성 순서

1) 테스트 계획 검토 및 자료 확보

2) 위험 평가 및 우선순위 결정

3) 테스트 요구사항 정의

4) 테스트 구조 설계 및 테스트 방법 결정

5) 테스트 케이스 정의

6) 테스트 케이스 타당성 확인 및 유지 보수

3. 테스트 시나리오

1) 테스트 케이스를 적용하는 순서에 따라 여러 개의 테스트 케이스들을 묶은 집합

2) 순서 존재, 구제적인 절차

4. 테스트 시나리오 작성 시 유의 사항

1) 여러 개의 시나리오로 분리하여 작성

2) 유스케이스 간 업무 흐름이 정상적인지 테스트 가능해야 함

5. 테스트 오라클

1) 테스트 결과가 올바른지 사전에 정의된 참 값을 대입하여 비교하는 것

2) 예상 결과를 계산하거나 확인

3) 특징

- 제한된 검증 : 모든 테스트 케이스에 적용할 수 없다.

- 수학적 기법 : 수학적 기법을 이용하여 구할 수 있다.

- 자동화 기능 : 실행,비교 등을 자동화 할 수 있다.

4) 종류

- 참 오라클 : 모든 케이스의 입력 값에 대해 기대하는 결과를 제공하는 오라클, 모든 오류 검출

- 샘플링 오라클 : 특정 몇 개만 입력

- 추정(Heuristic) 오라클 : 특정 몇 개만 입력 후 나머지 값에 대해 추정으로 처리

- 일관성 검사 오라클 : 변경이 있을 때 테스트 케이스의 수행 전과 수행 후 값이 동일한지 확인

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

61주차 - 테스트 자동화 도구

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 테스트 자동화의 개념

1) 테스트 절차를 스크립트 형태로 구현하는 자동화 도구 적용

2) 휴먼 에러 줄이고 정확성,품질 향상

2. 테스트 자동화 도구의 유형

1) 정적 분석 도구(Static Analysis Tools)

- 프로그램 실행하지 않고 분석하는 도구, 소스코드에 대한 결함 발견

- 테스트 수행하는 사람이 코드를 이해하고 있어야만 가능

2) 테스트 케이스 생성 도구

- 자료 흐름도 : 자료 원시 프로그램을 입력받아 파싱 후 자료 흐름도 작성

- 기능 테스트 : 주어진 기능을 구동시키는 모든 상태를 파악하여 입력을 작성

- 입력 도메인 분석 : 원시 코드 내부 참조하지 않고, 입력 변수의 도메인 분석하여 테스트 데이터 작성

- 랜덤 테스트 : 입력 값 무작위로 추출하여 테스트

3) 테스트 실행 도구

- 스크립트 언어를 사용

- 데이터 주도 접근 방식

- 데이터 저장하고 스크립트에 익숙하지 않은 사용자도 가능

- 키워드 주도 접근 방식

- 수행 동작 키워드와 데이터 저장하여 실행, 키워드를 이용하여 테스트 정의

4) 성능 테스트 도구

- 가상의 사용자를 만들어 테스트 수행

5) 테스트 통제 도구

- 형상관리도구, 결함추적관리도구

6) 테스트 하네스 도구

- 테스트를 지원하기 위해 생성된 코드와 데이터

- 테스트가 실행될 환경 시뮬레이션, 컴포넌트,모듈이 정상적으로 테스트되도록 함

- 구성 요소

- 테스트 드라이버 : 하위 모듈 호출,매개변수 전달,결과 도출

- 테스트 스텁 : 타 모듈의 기능만 단순히 수행, 일시적 필요한조건 가지는 테스트용 모듈

- 테스트 슈트 : 테스트 케이스의 집합

- 테스트 케이스 : 테스트 항목

- 테스트 스크립트 : 실행 절차에 대한 명세서

- 목 오브젝트 : 사전에 사용자 행위를 조건부로 입력, 예정된 행위를 수행하는 객체

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

62주차 - 결함 관리

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 결함 관리 프로세스 순서(결함 계새기, 너검 수정이 재모)

1) 결함 관리 계획

2) 결함 기록

3) 결함 검토

4) 결함 수정

5) 결함 재확인

6) 결함 상태 확인 및 모니터링

7) 최종 결함 분석 및 보고서 작성

2. 결함 상태 추적

1) 결함은 지속적으로 상태 변화를 추적하고 관리해야 함

2) 결함 관리 측정 지표

- 결함 분포 : 모듈,컴포넌트의 특정 속성에 해당하는 결함 수 측정

- 결함 추세 : 진행 시간에 따른 결함 수 추이 분석

- 결함 에이징 : 특정 결함 상태로 진행되는 시간 측정

3. 결함 추적 순서(등신 검토 할 수 조차 없네 종료해)

1) 결함 등록

2) 결함 검토

3) 결함 할당

4) 결함 수정

5) 결함 조치 보류

6) 결함 종료

7) 결함 해제

4. 결함 분류

1) 시스템 결함 : 앱 환경이나 디비 처리에서 발생된 결함

2) 기능 결함 : 앱 기획,설계 등 단계에서 유입된 결함

3) GUI 결함 : 화면 설계에서 발생된 결함

4) 문서 결함 : 사용자 요구사항과 기능 요구사항의 불일치, 의사소통이 원활하지 않아 발생된 결함

5. 결함 심각도

1) high : 핵심 요구사항 미구현, 더이상 프로세스 진행 못하는 결함

2) medium : 부정확한 기능 시스템, 흐름에 영향을 미치는 결함

3) low : 부정확한 GUI 메세지, 시스템 흐름에는 영향을 미치지 않는 결함

6. 결함 우선순위

1) 심각도가 높다고 반드시 우선순위가 높은 것이 아님

2) critical,high,medium,low

7. 결함 관리 도구

1) Mantis : 작업 내용 기록, 결함 추적 가능

2) Trac : 결함 통합 관리

3) Redmine :

4) Bugzilla : 심각도 우선순위 지정

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

63주차 - 애플리케이션 성능 분석

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 애플리케이션 성능 지표

1) 처리량

2) 응답 시간

3) 경과 시간

4) 자료 사용률

2. 성능 테스트 도구

1) JMeter : HTTP 등 프로토콜 지원하는 부하 테스트

2) LoadUI : 서버 모니터링, 사용자 편의성 강화

3) OpenSTA : HTTP 등 프로토콜 지원하는 부하 테스트

4) 부하 테스트, 스트레스 테스트

- 부하 테스트 : 일정 시간 부하 가함

- 스트레스 테스트 : 과부하 상태에서 어떻게 동작

3. 시스템 모니터링 도구

1) 자원량 확인하고 분석하는 도구

2) 종류

- Scouter : 단일 뷰 통합/실시간 모니터링

- Zabbix : 웹기반 서버

4. 용어

1) 커넥션 풀 : 디비에 연결된 커넥션을 풀에 미리 만들어 놓고 필요할 떄 꺼내고 반환하는 기법

2) 연결 누수 : 풀에서 꺼낸 후 반납 안해서 풀이 줄어드는 현상

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

64주차 - 복잡도

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 복잡도 개요

1) 주요 측정 방법 : LOC(Line of Code), 순환 복잡도

2. 시간 복잡도

1) 점근 표기법 종류

- 빅오 표기법 : 실행시간 최악, 항상 최악보단 빠름

- 세타 표기법 : 실행시간 평균,

- 오메가 표기법 : 실행시간 최상, 항상 최상보다 느림

3. 순환 복잡도

1) 논리적인 복잡도 측정, 멕케이브 복잡도 메트릭, 제어 흐름도 이론에 기초를 둠

2) 영역의 수

3) V(G) = E - N + 2(E:화살표 수, N:노드 수)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

65주차 - 애플리케이션 성능 개선

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 소스 코드 최적화

1) 클린 코드

- 작성 원칙(가족단위의 중국추)

- 가독성

- 단순성

- 의존성 배제

- 중복성 최소화

- 추상화

2) 나쁜 코드

- 스파게티 코드 : 스파게티처럼 얽혀 있는 코드

- 외계인 코드 : 아주 오래되거나 참고문서 개발자가 없어 유지 보수 작업이 어려운 코드

멀리서 온 외계인이 만들어놓고 간 코드

2. 소스 코드 최적화 유형

1) 클래스 분할 배치 : 응집도 높임 크기 작게

2) 느슨한 결합 : 의존성 최소화

3) 코딩 형식 준수

4) 좋은 이름 사용

5) 적적할 주석문 사용

3. 소스 코드 품질 분석 도구

1) 정적 분석 도구

- 코드 분석 도구, 개발 초기 결함

- pmd : 미사용 변수, 최적화 되지 않은 코드

- cppcheck : C/C++ 메모리 누수,오버플로

- SonarQube : 중복 코드,복잡도

- checkstyle : 자바 코드에 대해 표준 검사

- ccm : 다양한 언어 코드 복잡도

- cobertura : 자바 언어 테스트 커버리지

2) 동적 분석 도구

- 코드 실행하여 결함 분석

- Avalanche : Valgrind 프레임워크 기반 구현, 취약점 분석

- Valgrind : 메모리 쓰레드 결함 분석

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

67주차 - 모듈 연계를 위한 인터페이스 기능 식별

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 모듈 연계의 개요

1) EAI(Enterprise Application Integration)

- 정보 전달 등 상호 연동이 가능하게 해주는 솔루션

- 종류

- Point-to-Point : 전부다 1:1 연결, 변경 재사용 어려움

- Hub & Spoke : 허브 가운데 두고 스포크 연결, 중앙 집중형, 허브 장애 시 전체에 영향

- Message Bus(ESB) : 가운데 버스 둠,앱 사이에 미들웨어 두어 처리, 대용량 처리

- Hybrid : hub,bus 섞은 것, 데이터 병목 현상 최소화

2) ESB(Enterprise Service Bus)

- 애플리케이션 간 연계,변환 표준 기반 인터페이스 제공해주는 솔루션

- EAI보단 서비스 중심 통합 지향

- 결합도 약하게 유지

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

70주차 - 인터페이스 구현

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 인터페이스 구현

1) 송수신 시스템 간의 데이터 교환 및 처리를 실현해 주는 작업

2. 데이터 통신을 이용한 인터페이스 구현

1) JSON(JavaScript Object Notation) : 속성-값 쌍, 자바스크립트

2) XML(eXtensible Markup Language) : 특수한 목적을 갖는 마크업 언어를 만드는데 사용됨

3) AJAX(Asynchronous JavaScript and XML) : 자바스크립트를 이용, 클라이언트와 서버 간 XML데이터, 비동기 통신 기술

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

72주차 - 인터페이스 보안

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 인터페이스 보안 기능 적용

1) 네트워크 영역

- 송수신 간 스니핑 방지하기 위해 네트워크 트래픽 암호화

- 스니핑 : 네트워크 중간에서 남의 패킷 정보 도청하는 수동적 공격

- IPSec : 네트워크 계층,패킷단위

- SSL : TCP/IP 계층 애플리케이션 계층 사이 인증

- S-HTTP : 클라이언트/서버 간 메세지 암호화

2) 애플리케이션 영역

3) 데이터베이스 영역

- 데이터베이스 동작 객체 보안 기능

2. 데이터 무결성 검사 도구

1) 송수신간 데이터가 변경됐는지 검사, 알려줌

2) Tripwire, AIDE, Samhain, Claymore, Slipwire, Fcheck

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

74주차 - 인터페이스 구현 검증

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 인터페이스 구현 검증 도구

1) xUnit : 같은 코드 여러번 작성하지 않게함, Smalltalk

2) STAF : 서비스 호출 및 컴포넌트 재사용 등 다양한 환경 지원, 데몬이 응답 대신

3) FitNess : 웹 기반 테스트 설계,실행,확인

4) NTAF : FitNess장점 STAF 장점 결함, NHN(Naver)

5) Selenium : 웹 앱 테스트

6) watir : 루비 사용 앱 테스트