객체기술에 의한 소프트웨어 테스팅 기술

이농용

바산네트워크㈜ 대표이사
숭실대학교 컴퓨터학부 교수

목차

Part I

Software Testing Process

목차

테스트란 무엇인가?

테스트의 목적

- 테스트의 목적은 소프트웨어 개발 시 달성해야 하는 품질에 대해 평가하고 문제점을 발견하는 것
  - 모든 요구사항이 올바르게 구현되었는가를 검증
  - 소프트웨어 배포 이전에 결함 및 오류의 해결
- 테스트 워크플로우는 기타 워크플로우와 다음과 같은 관계를 가짐
  - 요구사항 워크플로우 : 테스트 요구사항 수집
  - 분석&설계 워크플로우 : 시스템 설계 방법 결정
  - 구현 워크플로우 : 테스트 대상이 되는 구현모델 생성
  - 환경 워크플로우 : 테스트 지원 위한 산출물을 생성, 유지보수
  - 관리 워크플로우 : 전체 프로젝트의 반복 계획

테스트의 분류

- 품질 차원에 따른 분류
- 테스트의 단계에 따른 분류
- 테스트의 유형에 따른 분류

7

품질 차원에 따른 분류

- 신뢰성(Reliability)
  - MTTF (Mean Time To Failure), 소프트웨어 코드의 무결성(Integrity), 구조(Syntax 준수여부), 견고성(메모리 누수, 실패에 대한 결고성)에 대한 테스트
- 기능(Function)
  - 요구사항에 명시된 대로 수행되는가를 테스트
- 시스템 성능(System Performance)
  - 실제 운영시의 부하 상에서 응답시간, 한계 부하치 등을 테스트

8

테스트의 단계에 따른 분류

- 테스트는 작은 단위에 대한 테스트(Unit test)에서부터 전체 시스템에 대한 테스트(System test)로 진행
  - 단위 테스트 단계(Unit test Stage): 시스템에서 테스트 가능한 가장 작은 단위를 개별적으로 테스트
  - 통합 테스트 단계(Integration test Stage): 컴포넌트나 서브시스템과 같은 통합 단위를 테스트
  - 시스템 테스트 단계(System test Stage): 전체 응용프로그램 또는 시스템을 테스트
  - 인수 테스트 단계(Acceptance test stage): 배포여부를 결정하기 위해 최종 사용자에 의해 수행되는 완전한 시스템에 대한 테스트

9

테스트의 유형에 따른 분류

- 벤치마크 테스트(Benchmark test)
  - 널리 알려진 워크로드 또는 시스템과 새로 설계한 테스트 대상의 성능 비교
- 구성 테스트(Configuration test)
  - 다양한 하드웨어와 소프트웨어 구성하에서 테스트 대상이 제대로 동작하는가 검증
- 기능 테스트(Function test)
  - 테스트 대상이 요구되는 기능을 제대로 수행하는가 검증
- 설치 테스트(Installation test)
  - 다양한 하드웨어와 소프트웨어 구성하에서 테스트 대상이 제대로 설치되는가를 검증
- 무결성 테스트(Integrity test)
  - 프로그래밍 언어의 문법과 구문을 준수했는가, 시스템 자원을 적절히 사용했는가와 같은 코드의 신뢰도 및 코드의 견고성을 검증

10

테스트 유형에 따른 분류(계속)

- 부하 테스트(Load test):
  - [illegible]
  - [illegible]
- 퍼포먼스 테스트(Performance test)
  - [illegible]
- 스트레스 테스트(Stress test)
  - [illegible]
- 볼륨테스트(Volume Test)
  - [illegible]
- 회귀 테스트(Regression test)
  - [illegible]
  - [illegible]
  - [illegible]

11

테스트 생명주기

- UP에서의 소프트웨어의 생명주기
  - 반복(Iteration)이 진행됨에 따라 점차 개선됨
  - 테스트의 생명주기 역시 반복됨
    - 반복마다 릴리즈가 산출되면 이를 테스트함
    - 작성된 스크립트는 다음 반복에서도 회귀 테스트(Regression Test)를 위해 사용됨
    - 반복적인 접근법은 테스트 스크립트의 부분적 수정을 요구하기도 함

12

테스트의 단계

- 작은 단위 테스트에서부터 전체 시스템에 대한 테스트로 진행됨
  - 단위 테스트 (Unit Test)
    - 반복 초기에 구현
    - 일반적으로 개발자가 구현한 컴포넌트에 적용
  - 통합 테스트 (Integration Test)
    - 컴포넌트 통합 시 유스케이스에 명시된 기능 수행하는지 검증
    - 하나의 패키지 또는 몇 개의 패키지 묶어 이를 대상으로 수행
  - 시스템 테스트 (System Test)
    - 소프트웨어가 전체로서 기능할 때 또는 전체 중 주요 부분이 구현되었을 때 수행
    - 전체 시스템을 대상으로 하는 것을 원칙으로 함

성능 테스트

- 시스템의 응답시간, 동작의 신뢰성, 처리 한도 등과 같은 성능에 관련된 특성을 평가
- 각기 다른 목적을 가진 여러 가지 유형의 테스트를 포괄적으로 의미
  - 벤치마크 테스트(Benchmark Test)
  - 퍼포먼스 테스트(Performance Test)
  - 부하 테스트(Load Test)
  - 스트레스 테스트(Stress Test)
  - 볼륨 테스트(Volume Test)

구조 테스트

- 웹 기반의 응용 프로그램은 일반적으로 정적/동적의 연결(Link)를 포함하기 때문에 구조적으로 취약할 수 있음
- 구조 테스트에서는 이러한 연결의 적합성 여부를 검증하는 테스트 수행
  - 모든 연결에 대해서 적절한 내용이 표시되는가를 검증
  - 손상된 연결(Broken Link)이 없는가 확인
  - 연결되지 않는 내용(Orphaned Content)이 있는가 확인

인수 테스트

- 소프트웨어의 배포 전에 수행하는 최종 테스트
- 소프트웨어가 최종 사용자에 의해 사용될 준비가 되었는가를 검증하는 것이 주 목적
- 세가지 유형의 전략을 채택할 수 있음
  - 공식적인 인수 테스트(Formal Acceptance Test)
  - 비공식적인 인수테스트(Informal Acceptance Test)
  - 베타 테스트(Beta Test)

테스트의 측정 기준

- 커버리지(Coverage)
  - 테스트의 완전성(Completeness)에 대한 측정기준
    - 전체의 몇 % 가 테스트 되었는가를 나타냄
  - 코드에 대한 커버리지, 테스트 항목에 대한 커버리지, 테스트 요구사항에 대한 커버리지 등 테스트의 커버리지로 표현
- 품질(Quality)
  - 시스템의 신뢰성(Reliability), 안전성(Safety), 성능(Performance)의 측정기준
  - 테스트 중 발견한 결함(Defect)에 대한 분석 통해 표현

테스트 전략

- 테스트 전략 : 테스트 관련 액티비티(Activity)의 일반적인 접근법과 목적에 대해 설명
  - 채택된 테스트 기법과 도구를 기술
  - 테스트 종료조건 및 성공으로 평가하기 위한 조건 제시

목차

Part I – Software Testing Process

- 테스트의 개요 (Introduction to Test)
- 테스트 워크플로우 (Test Workflow)
- 테스트 워크플로우의 산출물 (Artifacts of Test Workflow)

Part II – Software Testing with Object Oriented Technology

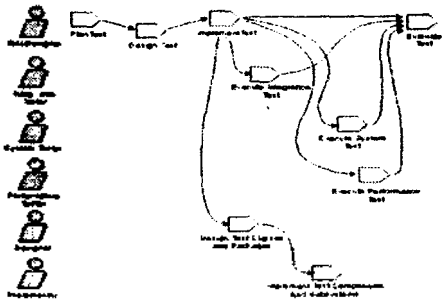
- 구조적 프로그램 시험방법과 객체지향 프로그램 시험방법의 비교
- 결과 지향 시험 전략 (Result-Oriented Test Strategy)
- 재사용을 고려한 객체지향 프로그램의 시험

결론

테스트 워크플로우

- 테스트 워크플로우의 구성
  - 테스트 계획 및 설계(Plan and Design Test)
    - 아키텍처 구축 단계에서 수행
    - 구축할 소프트웨어에 아키텍처와 설계가 진행됨에 따라 테스트에 대한 계획과 설계 작업이 수행
  - 테스트 구현 (Implementation Test)
    - 테스트의 설계를 기반으로 하여 테스트 스크립트를 구현
  - 테스트 수행 및 평가 (Execution and Evaluation Test)
    - 각각의 반복마다 실행 가능한 빌드가 작성되면 구현된 테스트 스크립트를 이용하여 테스트를 수행하고 결과를 평가

테스트 워크플로우(계속)



테스트 워크플로우의 작업자

- **테스트 설계자(Test Designer)**
  - 테스트를 계획하고, 설계하고, 구현하고 테스트의 결과를 평가
- **시스템 테스트 담당자(System Tester)**
  - 시스템 테스트를 담당. 테스트 환경 (테스트 스크립트, 하드웨어 등)을 설정, 테스트 수행을 평가, 오류 처리, 테스트 결과의 평가, 발견된 결함 보고서 등의 작업을 수행
- **성능 테스트 담당자(Performance Tester)**
  - 성능 테스트를 담당
- **통합 테스트 담당자(Integration Tester)**
  - 통합 테스트를 담당
- **구현 설계자(Implementer)**
  - 테스트를 위한 driver나 stub같은 특별한 프로그램 (스크립트 등)의 구현을 수행
- **설계자(Designer)**
  - 테스트를 위한 driver나 stub같은 특별한 프로그램 (스크립트 등)의 설계를 담당

테스트 워크플로우의 산출물

- **테스트 계획(test plan)**
  - 테스트의 목적에 대한 정보를 수록
- **테스트 모델**
  - 테스트 항목과 테스트 절차, 그리고 테스트 스크립트로 구성
- **워크로드(workload) 모델**
  - 성능평가를 위해 사용
- **결함(defects)**
  - 실패한 테스트로부터 생성되고 하나의 변경요구로 등록
- **테스트를 위한 패키지와 클래스, 테스트를 위한 서브시스템 (Subsystem)과 컴포넌트(Component)**

테스트 워크플로우의 세부 항목

- **테스트 계획 및 설계**
  - 시스템 테스트 계획 및 설계(Plan and Design System Test)
  - 통합 테스트 계획 & 설계(Plan and Design Integration Test)
  - 성능 테스트 계획 & 설계(Plan and Design Performance Test)
- **테스트 구현**
  - 시스템 테스트 구현(Implement System Test)
  - 통합 테스트 구현(Implement Integration Test)
  - 성능 테스트 구현(Implement Performance Test)
- **테스트 수행**
  - 시스템 테스트 수행(Execute System Test)
  - 통합 테스트 수행(Execute Integration Test)
  - 성능 테스트 수행(Execute Performance Test)
- **테스트 평가**
  - 시스템 테스트 평가(Evaluate System Test)
  - 통합 테스트 평가(Evaluate Integration Test)
  - 성능 테스트 평가(Evaluate Performance Test)

목차

Part I – Software Testing Process

- 테스트의 개요 (Introduction to Test)
- 테스트 워크플로우 (Test Workflow)
- 테스트 워크플로우의 산출물 (Artifacts of Test Workflow)

Part II – Software Testing with Object Oriented Technology

- 구조적 프로그램 시험방법과 객체지향 프로그램 시험방법의 비교
- 결과 지향 시험 전략 (Result-Oriented Test Strategy)
- 패턴을 적용한 객체지향 프로그램의 시험

결론

25

테스트 워크플로우의 산출물

- 테스트 지침 (Test Guidelines)
- 테스트 계획 (Test Plan)
- 워크로드 모델 (Workload Model)
- 테스트 모델 (Test Model)
- 테스트 프로시듀어 (Test Procedure)
- 테스트 케이스 (Test Case)
- 테스트 스크립트 (Test Script)
- 결함 (Defect)

26

테스트 지침

- 반복마다 테스트 전략을 어떻게 정의할 것인가에 대해 기술
- 테스트 지침의 구성요소
  - 개요 : 테스트 지침의 목적 및 전략을 기술
  - 참고문헌
  - 테스트의 목표
  - 테스트 표준 : 테스트의 계획, 설계, 구현, 평가에서 사용될 지침과 표준에 대해 기술
  - 주요 측정방법 : 테스트의 진척도를 파악하기 위해 어떤 측정방법 사용할 것인가를 정의
  - 테스트 완료 조건 : 테스트 완료 조건과 평가 기준을 명시
  - 결함 관리 지침
  - 변경 관리 조건 : 테스트의 산출물을 어떻게 관리하고 유지보수할 것인가를 명시

27

테스트 계획

- 테스트의 목적, 목표, 전략, 자원에 대한 정보를 포함
- 테스트의 목적을 다음과 같은 모든 이해 당사자들(Stakeholders)이 공유할 수 있게 함
  - 최종 사용자 대표(End User Representative)
  - 고객(Customer)
  - 시스템 통합 담당자(System Integrater)
  - 테스트 설계자(Test Designer)

28

테스트 계획(계속)

- 테스트 계획의 구성요소
  - 개요 : 테스트의 목적, 배경, 범위에 대해 기술
  - 테스트 요구사항 : 테스트의 대상이 되는 유스 케이스(Use Case)와 요구사항 (Requirement)을 식별, 각각의 테스트할 요구사항에 대해 테스트의 위험요인과 우선순위에 대해 기술
  - 테스트 전략 : 수행할 테스트의 유형과 각각의 테스트의 목적을 기술, 테스트의 방법과 조건, 기타 고려사항에 대해서도 기술
  - 자원 : 테스트를 위해 필요한 자원을 기술
  - 일정/이정표
  - 테스트 산출물 : 테스트 중 생성되는 다양한 산출물과 보고서에 대해 기술

29

워크로드 모델

- 성능 테스트에서 사용되는 변수를 기술하고, 변수에 할당할 값을 정의
- 워크로드 모델과 관련된 작업자
  - 테스트 설계자(Test Designer) : 워크로드 모델을 통해 실제 운영에서와 같은 부하, 데이타, 최종 사용자가 요구하는 기능을 가장 잘 반영하는 유스 케이스와 흐름과 조건에 대해 기술
  - 성능 테스트 담당자(Performance Tester) : 워크로드 모델을 통해 성능 테스트의 목적을 이해하고, 성능 테스트를 위한 테스트 항목을 작성하고 구성, 수행
  - 최종 사용자 대표(User Representative) : 워크로드 모델의 내용을 검토하여 테스트 항목을 승인
- 프로젝트 초기의 초기 테스트 설계단계에서 초기 워크로드의 모델이 작성되며, 이후 반복을 거쳐 개선되고 상세해짐

30

워크로드 모델(계속)

- 워크로드 모델의 구성요소
  - 개요 : 성능테스트의 목적, 배경, 목표 기술
  - 시스템 속성과 변수 : 워크로드 모델의 대상이 되는 시스템의 고유한 특징을 나타내는 속성과 변수에 대해 기술
  - 부하 레벨 : 성능 테스트시 사용레이트와 클라이언트의 수를 기술
  - 액터 클래스 정의 : 시스템으로의 부하를 시뮬레이트 할 때 사용되는 유스 케이스의 액터를 대해 기술
  - 측정기준과 조건 : 성능 테스트를 위한 측정기준과 테스트의 성공여부를 가리기 위한 기준에 대해 기술
  - 사용자 속성 및 변수 : 워크로드 모델의 대상이 되는 시스템을 사용하는 최종사용자의 각각 다른 특성을 나타내는 속성과 변수에 대해 기술
  - 액터 클래스 작업 프로파일 : 액터에 의해 수행되는 특정 유스 케이스와 이의 흐름, 그리고 유스 케이스가 수행되는 시간에 대해 기술
  - 액터 클래스 프라이트 : 단위별 속도, 생각하는 시간과 같이 사용자와 시스템간의 상호작용에서 수반되는 사용자마다 고유한 값에 대해 기술
  - 환경 리허설 테스팅에서 요구사항 : 성능 테스트를 위한 환경 구축시 발생하는 요구사항과 제약조건에 대해 기술

테스트 모델

- 테스트 케이스(Test Case)와 테스트 프로시듀어(Test Procedure)와 어플간의 관계를 증명하는 질문
- 테스트 모델과 관련 있는 작업자
  - 테스트 설계자(Test Designer) : 테스트 모델을 통해 테스트의 범위(scope)와 테스트로 목록이 어떻게 테스트 절차를 통해 구현되는가를 기술
  - 통 테스트 담당자(Integration Tester), 시스템 테스트 담당자(System Tester), 성능 테스트 담당자(Performance Tester) : 테스트 모델을 통해 테스트의 목표를 이해
  - 설계자(Designer) : 테스트를 위한 기능의 설계를 위해 사용
  - 구현 담당자(Implementer) : 테스트를 위한 기능의 구현을 위해 사용

테스트 모델(계속)

- 테스트 모델의 구성요소
  - 테스트 프로시듀어 : 특정한 테스트 항목(들)을 위한 테스트의 설정, 수행, 결과 평가에 대해 기술한 일련의 구체적인 명령
  - 테스트 케이스 : 특정한 목적을 위해 개발한 테스트의 입력, 실행 조건, 그리고 예상 결과의 모음
  - 테스트 스크립트 : 테스트 프로시듀어(전체 또는 일부)의 수행을 자동화하는 컴퓨터에 의해 수행 가능한 일련의 명령
- 특정 테스트 항목(들)을 위한 테스트의 설정, 수행, 결과 평가에 대해 기술한 일련의 구체적인 명령
- 테스트 담당자가 테스트 환경을 설정하고 테스트 케이스를 적절하게 구현하고 수행하기 위해 필요한 정보를 수록

테스트 프로시듀어

- 테스트 프로시듀어의 구성요소
  - 단계(Step)/액션(Action) : 테스트 항목을 수행할 때 액터가 실행하는 일련의 단계/목선을 기술
  - 입력 값(Input Values)/테스트 항목(Test Case) : 테스트 절차의 각 단계를 수행했을 때의 예상 결과를 기술
  - 검증 방법(Verification Methods) : 예상 결과와 실제 결과를 비교할 때 사용하는 방법
- 특정한 목적을 위해 개발한 테스트의 입력, 수행 조건, 그리고 예상 결과의 모음
- 유스 케이스가 적절하게 구현되었는가를 검증하기 위해 필요한 조건에 대해 기술

테스트 케이스

- 테스트 케이스의 구성요소
  - 테스트 케이스 설명(Test Case Description) :조건, 프로그램 수행 경로, 목표 등을 기술
  - 테스트 입력(Test Inputs) : 액터와 상호작용하는 객체 또는 변도, 그리고 테스트 항목을 수행할 때 여기에 액터가 입력할 데이터 값을 기술
  - 예상 결과(Expected Results) : 테스트 항목의 수행 종료시의 데이터 또는 상태를 기술

테스트 스크립트

- 테스트 스크립트는 테스트 절차의 수행을 자동화하는 컴퓨터에 의해 수행 가능한 일련의 명령
- 테스트 설계자에 의해 작성되어 테스트 절차를 효과적인 방법으로 수행하는 것을 목적으로 함
- 테스트 스크립트의 구성요소
  - 명령(Instructions) : 액터의 유스 케이스 수행을 에뮬레이트할 수 있는 컴퓨터에 의해 수행 가능한 명령
  - 검증 지점(Verification Point) : 실제 결과값이 예상 결과값과 비교되는 지점
  - 검증 방법(Verification Method) : 예상 결과와 실제 결과의 비교를 수행하는 컴퓨터에 의해 수행 가능한 명령

결함

- 소프트웨어 개발주기의 초기 단계에서 발견되는 불완전 항부터 충분하게 개발되어 테스트되거나 운용중인 소프트웨어에서 발견되는 오류의 징후까지 모두 망라하는 개념
- 모든 작업자에 의해서 작성될 수 있음
- 문제점에 대한 정보를 공유함으로써 문제점을 올바르게 해결하기 위해 기술

37

결함(계속)

- 결함의 구성 요소
  - 설명(Description) : 결함에 대한 간략한 기술
  - 우선순위(Priority) : 결함의 해결 우선순위를 기술
  - 중요도(Severity) : 결함이 응용프로그램에 미치는 영향을 기술
  - 결함 정보(Origin Data) : 어떻게 결함이 발견되었는가를 기술. 테스트 환경과 테스트 담당자 정보 포함
  - 상태(Status) : 결함의 현재 상태(Open, Fixed, Closed ..)를 기술
  - 해결(Resolution) : 결함을 어떻게 해결했는가를 기술. 어떤 변경이 필요했으며 소프트웨어가 어떻게 변경되었는지, 그리고 누가 해결했는지를 기술

38

Part II

Software Testing
with Object Oriented Technology

목차

40

구조적 프로그램 시험방법과 객체지향 프로그램 시험방법의 비교

- 캡슐화(Encapsulation)
- 상속(Inheritance)
- 다형성(Polymorphism)

41

캡슐화(Encapsulation)

- 직접적인 오류 발견을 방해
- 시험을 수행하기 위해서는 객체의 직접적이고도 정확한 상태 보고가 요구되나, 객체지향 언어는 이러한 기능 구현이 어려움

42

상속(Inheritance)

- 상속은 캡슐화를 약화시킴으로 인하여, 절차지향 언어의 전역 데이터처럼 오류를 유발할 수 있음
- 넓고 깊이 이루어지는 상속은 프로그램 이해에 방해가 되며, 오류의 요인이 되고, 시험의 원활한 수행이 어려워짐

43

다형성(Polymorphism)

- 객체가 표현하는 모든 경우에 대하여 시험이 진행되어야 함
  - 숨겨진 오류가 있을 수 있으며, 찾기 어려움
  - 하나의 시험에 대하여 시스템이 이상 없이 작동한다 하더라도, 안정성을 보장하지는 않음

44

목차

45

결과 지향 시험전략 (Result-Oriented Test Strategy)

- 결과 지향 시험전략이란 책임 기반의 테스트 디자인(responsibility-based test design)과 구현 기반의 테스트 디자인 (implementation-based test design)을 병합하여 보다 효과적인 테스트를 수행하는 방법

46

책임 기반의 시험 디자인 (Responsibility-based Test Design)

- 각 유닛, 서브 시스템 또는 시스템에 대한 시험 디자인을 할 때 기존에 명시되어 있거나 예상하고 있는 책임을 적용하여 시험을 수행하는 방법
- 각 유닛의 책임은 일반적으로 기능(Functionality)의 형태로 나타남
- 블랙박스 테스트가 대표적 접근 방법

47

구현 기반의 시험 디자인 (Implementation-based Test Design)

- 시험 케이스 개발을 위하여 소스를 분석하는 방법
- 구조적인 접근을 하며 화이트 박스 테스트가 대표적인 접근 방법

48

객체지향 프로그램 시험방법

- 객체지향 프로그램을 시험하여야 하는 경우에는 기존의 방법과 더불어 객체의 특성에 맞는 시험방법을 추가하여야 함
- 각 시험에 맞는 시험패턴을 시험의 목적에 따라 기존 시험에 추가하여 객체지향 프로그램을 시험할 수 있음

목차

패턴을 적용한 객체지향 프로그램 시험

- 단위 시험
  - 메서드 범위 시험
  - 클래스 범위 시험
  - 클래스 간 시험
- 통합 시험
- 시스템 시험

단위 시험

- 메서드 범위 시험디자인 패턴
  - 기본적 메서드 범위 시험 프로시듀어 (Basic Method Scope Test Procedure)
    - 모든 메서드 범위 시험의 공통적 프로시듀어
  - 카테고리 분류 시험(Category-Partition Test)
    - 도출된 값을 카테고리 기준으로 분류하는 방법
  - 복합기능 시험(Combinational Function Test)
    - 복잡한 논리 알고리즘 비즈니스 룰 등을 구현한 알고리즘을 시험하는 방법
  - 재귀기능 시험(Recursive Function Test)
    - 재귀호출 시험
  - 다형 메시지 시험(Polymorphic Message Test)
    - 상속된 클래스간의 오버라이딩(Overriding)을 시험

단위 시험

- 클래스 범위 시험디자인 패턴
  - Invariant Boundaries Test
    - 입력 데이터의 범위 내외의 값을 입력하여 정상적인 결과가 도출되는가를 시험하는 방법
  - Nonmodal Class Test
    - 메시지 시퀀스에 제약사항(Constraints)이 존재하지 않는 클래스 시험
  - Quasi-modal Class Test
    - 클래스의 상태가 변할 때 메시지 시퀀스의 제약이 항상 변하는 클래스를 시험할 시 사용
  - Modal Class Test
    - 정해진 메시지 시퀀스 제약이 존재하는 클래스를 시험할 때 사용

단위 시험

- 클래스간의 시험디자인 패턴
  - 다형 서버 시험 (Polymorphic Server Test)
    - 클래스 사이에서 상속이 구현되었을 경우, 서버 클래스의 모든 메서드가 올바르게 상속되는가를 테스트함
    - 서버 클래스에 있는 메서드의 리스트를 작성하고, 상속받은 클라이언트 클래스에서 한번씩 실행시켜서 모두 실행되는가를 테스트
  - 형식 조직 시험 (Modal Hierarchy Test)
    - 클래스에서 시행되었던 형식 클래스 시험을 클래스간에 적용한 시험

통합 시험

- 빅뱅 통합 (Big Bang Integration)
  - 모든 컴포넌트를 한 번에 시험하여 시스템의 안정성(Stability)를 시험하는 방법
- 상향식 통합 (Botton-up Integration)
  - 사용 독립성 (interleave dependency) 순서에 따라서 시험을 시행하는 방법이다
- 하향식 통합 (Top-down Integration)
  - 응용프로그램 관리 체계(application control hierarchy) 순서에 따라 시험을 시행하는 방법이다
- 협력 통합 (Collaboration Integration)
  - 협력(collaboration)과 각각의 독립성에 따라 통합의 순서를 결정하는 방법이다

56

통합 시험(계속)

- 백본 통합 (Backbone Integration)
  - 상향식 통합, 하향식 통합, 빅뱅 통합을 혼합하여 시험하는 방법이다 반복 개발을 지원한다 강한 결합(tightly coupled)으로 이루어진 서브시스템 간의 상호운용성 (Interoperability)를 확인하는데 목적이 있다
- 계층 통합 (Layer Integration)
  - 계층적 구조(Layered Architecture)가 ... 컴포넌트 및 인터페이스를 점증적으로 시험하는 방법이다
- 클라이언트/서버 통합 (Client Server Integration)
  - 중앙 서버 컴포넌트들은 약한 결합(loosely coupled) 클라이언트 컴포넌트를 시험하는 방법이다
- 분산 서비스 통합 (Distributed Services Integration)
  - 피어 투 피어 (Peer-to-peer) 컴포넌트와 약한 결합 네트워크 컴포넌트를 사용하는 ... 모든 노드가 시험될 때까지 점점 확장하여 시험한다

56

시스템 시험

- 확장 유스케이스 시험 (Extended Use Case Test)
  - 확장 유스케이스를 이용하여 시스템 시험을 시행한다 확장 유스케이스에는 다음과 같은 정보가 포함된다
    - 완벽한 실행 변수(operational variables)들의 목록
    - 각각의 실행 변수를 위한 범위 제약조건의 완벽한 명세(specification)
    - 각 유스케이스의 실행 관계(operational relation)
- CRUD 시험 (Coverd in CRUD)
  - CRUD 시험은 모든 기본 기능이 SUT(System Under Test) 안의 각각의 객체에서 수행되는가를 검사한다
- 프로파일에 따른 시험 할당 (Allocate Tests by Profile)
  - 모든 시험 자원을 각각의 유스케이스 상대 빈도에 따라 할당하는 시험방법이다 프로파일은 시스템의 사용 예(usage)를 나타낸다 자주 쓰는 기능일수록 더 많은 시험을 수행하여 시스템의 신뢰성(reliability)을 높일 수 있다

57

결론

- 과거 개발 프로세스에서의 테스트
  - 시스템 개발 완료 후 수행 또는 테스팅의 부재
  - 소프트웨어가 복잡화, 대형화로 인하여 효율적인 테스트 프로세스가 요구됨
- UP에서의 테스트 프로세스
  - 프로젝트의 초기부터 자동화된 테스트 반복적으로 수행
  - 소프트웨어 시스템의 결함 조기에 식별 및 제거
  - 자동화된 툴을 이용함으로써 생산성 향상
  - 뛰어난 품질의 소프트웨어를 적은 시간과 비용으로 구축 가능
- 객체기술에 의한 소프트웨어 테스팅
  - 객체지향 프로그래밍을 ... 객체지향의 특성을 고려하여, 구현된 특성에 맞는 시험을 ... 시험 절차로 추가하여 시행하여야 함
  - 객체지향 프로그램은 ... 다양한 다이어그램을 이용하여 시험 케이스를 작성할 수 있음

58