



강의노트

테스트

❖ 학습안내

이번 시간의 학습내용과 학습목표를 확인해보세요.

■ 학습내용

- 테스트의 단계
- 테스트의 유형
- 소프트웨어 검사

■ 학습목표

- 테스트의 단계를 설명할 수 있고, 활용할 수 있다.
- 테스트의 유형을 설명하고, 수행할 수 있다.
- 테스트를 통하여 소프트웨어 검사를 이해하고, 수행할 수 있다.



❖ 학습내용

[1] 테스트의 단계


1. 소프트웨어 테스트

- ◆ 소프트웨어 테스트의 정의
 - 노출되지 않은 숨어있는 결함(Fault)을 찾기 위해 소프트웨어를 작동시키는 일련의 행위와 절차로 **오류 발견을 목적으로 프로그램을 실행**하여 품질을 평가하는 과정
 - 테스트를 통하여 구축된 시스템이 검증됨
 - 테스트는 프로젝트의 **구현단계가 끝난 후**, 주어진 절차에 맞도록 수행하는 것이 일반적임

디버깅(Debugging)

이미 노출된 **소프트웨어의 결함을 없애는** 작업

◆ 소프트웨어 테스트의 목표

-  프로그램에 잠재된 **오류의 발견**
-  기술적인 **기능 및 성능의 확인**
-  사용자 **요구만족도, 제품신뢰도 향상**

◆ 소프트웨어 테스트의 특징

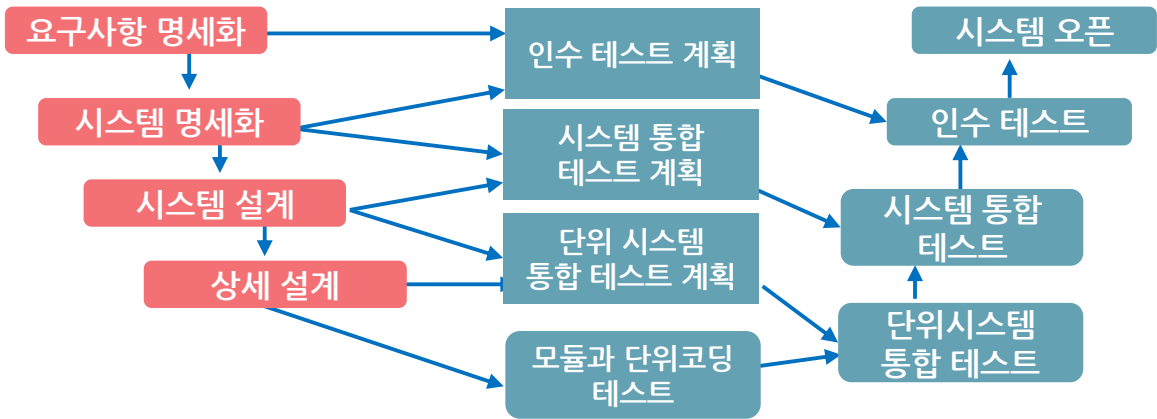
- 성공적인 테스트는 무결점이 아닌 **결함을 찾는데** 있음
- **테스트 케이스 선정, 테스트 계획 수립**에 따라 영향(미발견 결함을 발견하게 해줄 확률)
- 테스트 케이스는 기대되는 표준결과를 포함하여 **예측오류, 기대되지 않는 결함이 있다는** 가정하에 테스트계획 수립
- 개발자가 자기 프로그램을 **직접 테스트하지 않음** (디버깅 수행)
- 능력 있는 테스트 수행자는 성공적이고 효율적으로 시험을 수행

❖ 학습내용

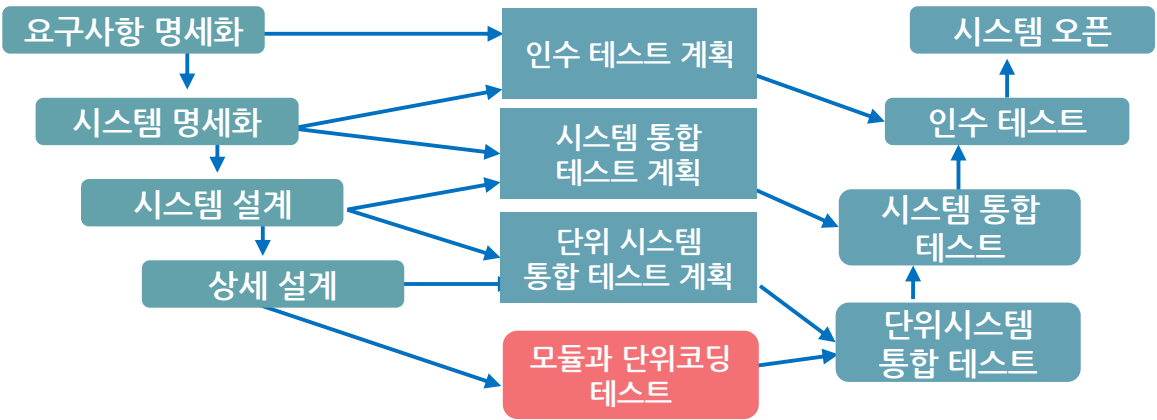
[1] 테스트의 단계

1. 소프트웨어 테스트(계속)

◆ 소프트웨어 개발 단계별 테스트



- 프로젝트에서의 진행절차의 일부
- 요구사항과 시스템에 대하여 명세서를 작성한 후, 이를 가지고 **시스템 설계**를 함
- 시스템 설계 후 **상세 설계** 작업이 이루어짐



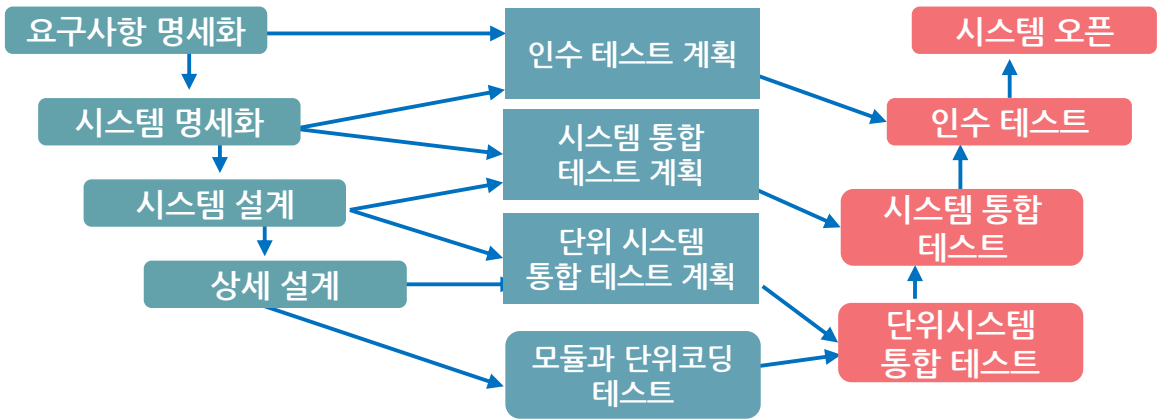
- 설계 후 개발 단계에서 개발단위마다 **모듈과 단위 코딩**에 대하여 테스트함

❖ 학습내용

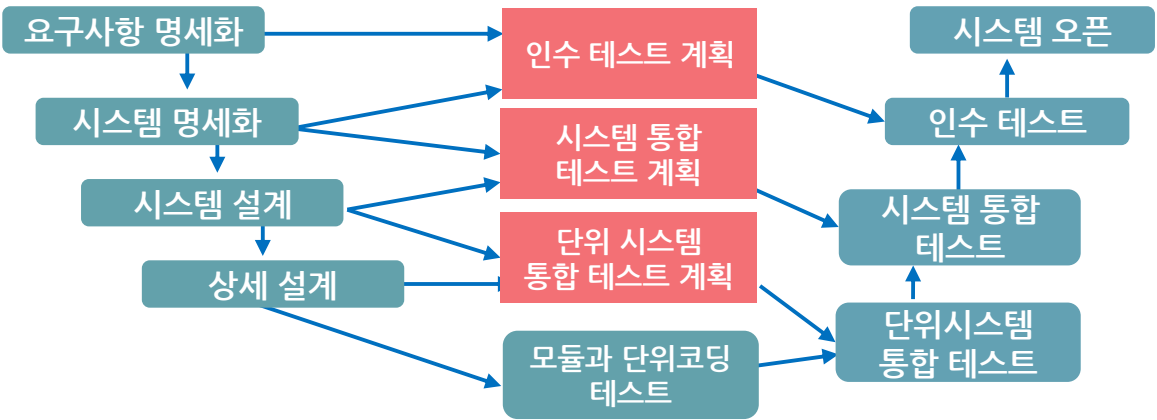
[1] 테스트의 단계

1. 소프트웨어 테스트(계속)

◆ 소프트웨어 개발 단계별 테스트(계속)



- 테스트 단계에서는 각각의 부분 시스템에 대하여 통합 테스트를 한 후, 시스템에 대하여 **통합 테스트**를 함
- 최종 인수자가 인수 테스트 수행 후 **시스템을 오픈**하게 됨



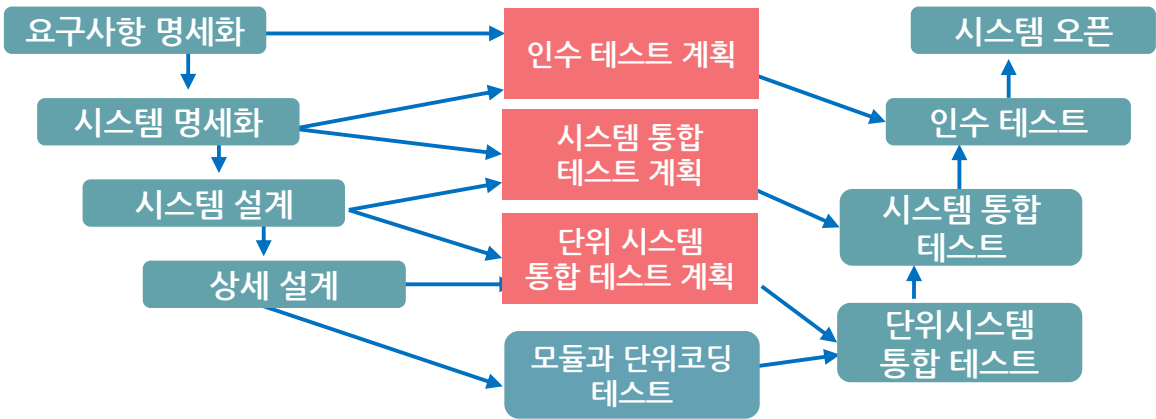
- 여기서 중요한 사항은 단위시스템 통합 테스트를 위한 계획은 시스템 **설계단계 끝부분과 상세설계 처음부분에 수행**함

❖ 학습내용

[1] 테스트의 단계

1. 소프트웨어 테스트(계속)

◆ 소프트웨어 개발 단계별 테스트(계속)



- 마찬가지로 시스템 통합 테스트와 인수 테스트도 이전 시스템명세화 단계와 시스템 설계단계에서 계획되어야 함

2. 단위 테스트, 통합 테스트

◆ 단위 테스트

- 단위 테스트 개요
 - 설계의 최소 단위인 **모듈**을 TEST함
 - **화이트박스 기법** 이용
- 단위 테스트 유형

유형	내용
인터페이스	다른 모듈과의 데이터 인터페이스 에 대하여 TEST
자료구조	모듈 내의 자료 구조상 오류 가 없는지를 TEST
수행경로	구조 및 루프 TEST 등에 의해 논리 경로 TEST
오류처리	각종 오류들이 모듈에 의해 적절히 처리여부 TEST
경계	오류가 발생하기 쉬운 경계 값 으로 TEST 사례

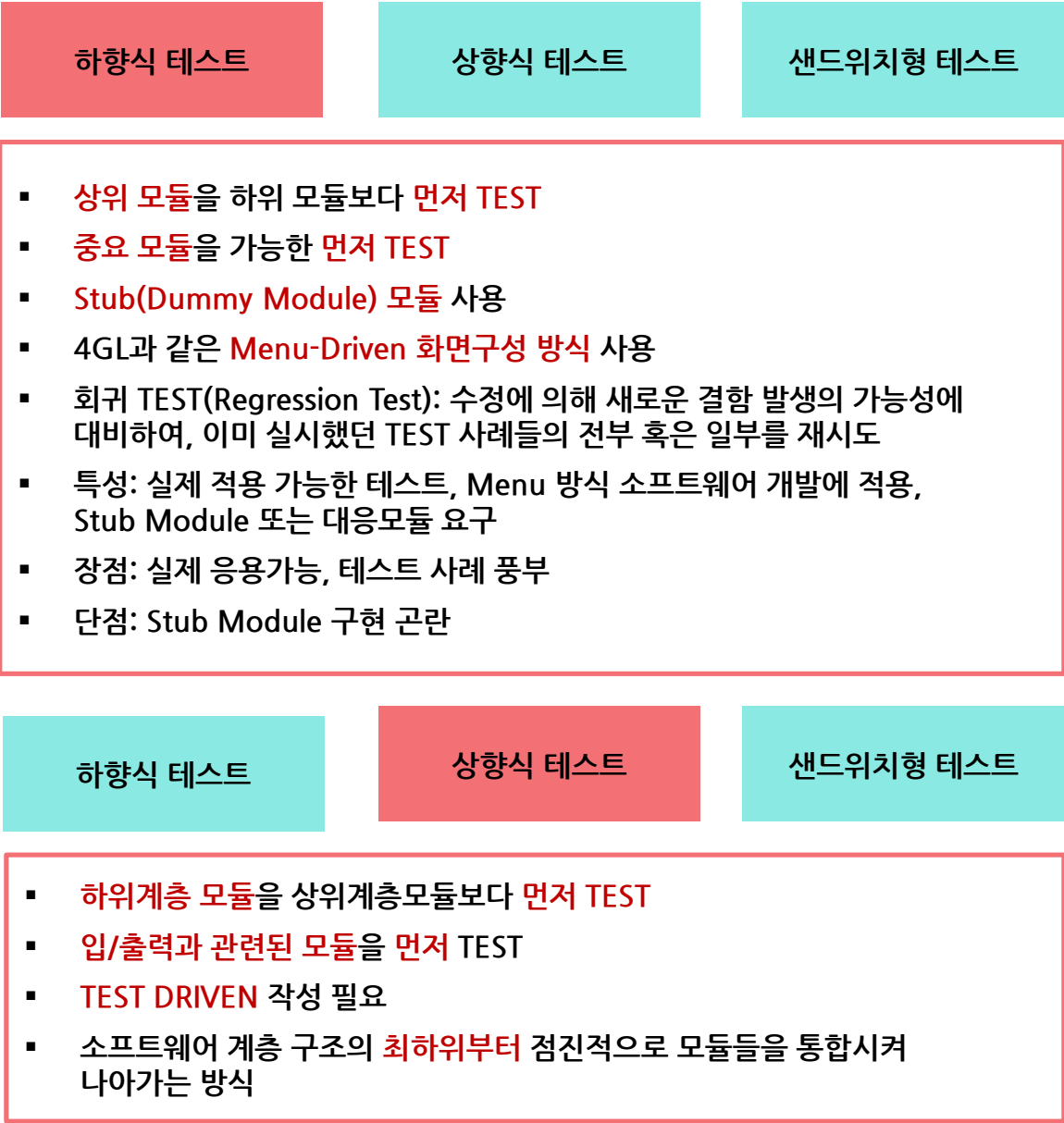
❖ 학습내용

[1] 테스트의 단계

2. 단위 테스트, 통합 테스트(계속)

◆ 통합 테스트

- 통합 테스트 개요
 - 단위 TEST을 거친 모듈들의 인터페이스(Interface)에 관한 오류발견이 목적
 - 모듈간의 체계적인 조합과정
 - 모듈들을 체계적으로 조합시킬 목적으로 모듈간의 인터페이스와 관련된 결함들을
 - TEST에 의해 발견하고 제거하는 작업
- 통합 테스트 유형



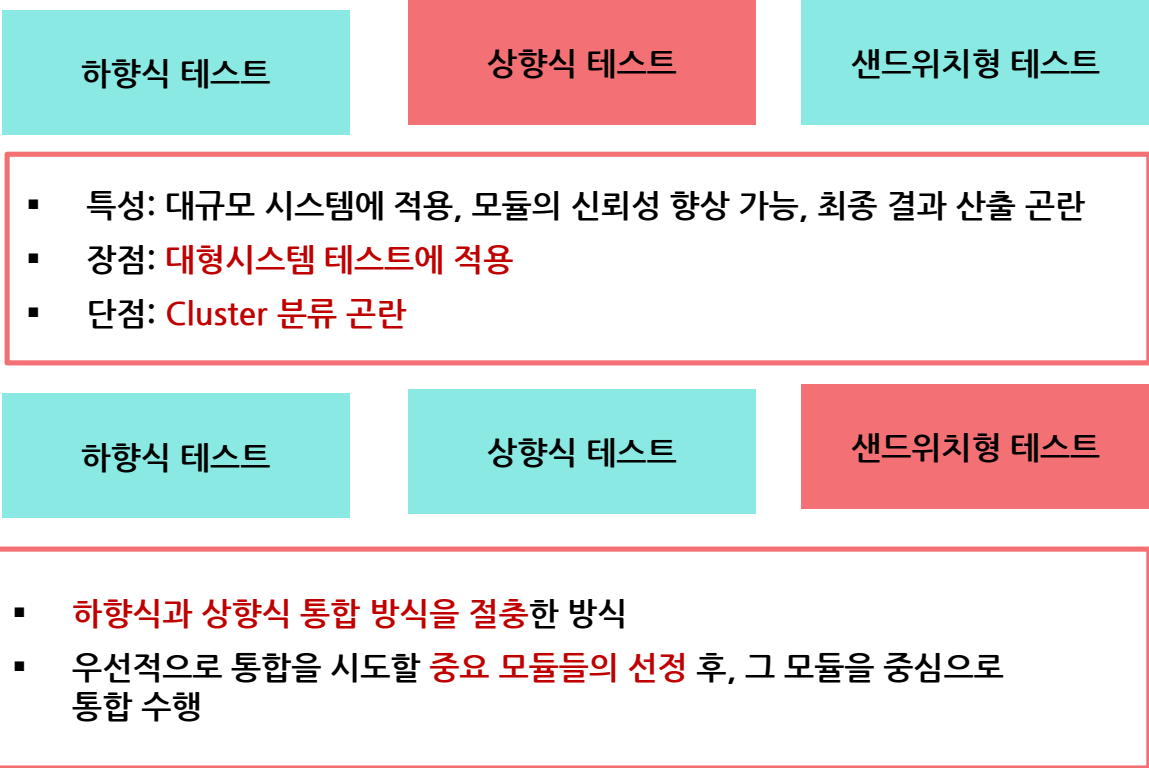
❖ 학습내용

[1] 테스트의 단계

2. 단위 테스트, 통합 테스트(계속)

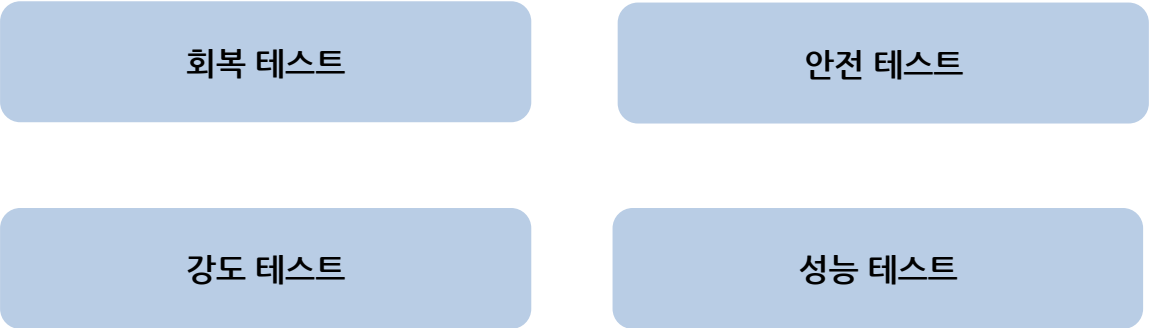
◆ 통합 테스트(계속)

- 통합 테스트 유형



◆ 시스템 테스트

- 사용자의 신뢰성을 확보하기 위하여 컴퓨터, 네트워크 제반 모든 사항에 관계된 테스트



❖ 학습내용

[1] 테스트의 단계

2. 단위 테스트, 통합 테스트(계속)

◆ 시스템 테스트(계속)

1 회복 테스트

- 소프트웨어가 다양한 방법으로 **실패하도록 유도**하고 **회복이 적절하게 수행**되는지를 검증하는 TEST
- 회복이 시스템에 의해 **자동**으로 수행되면 재 초기화, 데이터 회복, 재 시작 방법 등에 의해 정상적으로 회복되는지를 평가
- 운영체제, DBMS, 통신용 소프트웨어 등의 **완전성 TEST**

2 안전 테스트

- 시스템 내의 보호 기능이 **불법적인 침투**로부터 시스템을 **보호**하는지에 대한 검증 TEST
- 해커 등의 불법적 침입자로부터 **시스템의 보호 목적**으로 시행

3 강도 테스트

- 비정상적인 값, 양, 빈도의 자원의 입력에 대한 **정상 수행상태를 TEST**
- 소프트웨어에게 **다양한 스트레스**를 가해보는 TEST
- **민감도 TEST(Sensitivity Test)**: 유효한 입력 유형 중에서 불안정하게 하거나 부적절한 결과를 일으키는 데이터의 조합을 밝히도록 함

4 성능 테스트

- 통합시스템의 전·후 관계에서 **소프트웨어의 실행시간 TEST**
- 소프트웨어의 **효율성을 진단**하는 TEST
- 자원 이용, 처리시간, 요구된 응답 반응 등 **성능 TEST**

❖ 학습내용

[1] 테스트의 단계

2. 단위 테스트, 통합 테스트(계속)

◆ 인수 테스트

◆ 인수 테스트

- 사용자측 관점에서 소프트웨어가 요구사항을 충족시키는지 평가
- 소프트웨어가 고객의 합리적인 기대에 따라 제 기능을 발휘하는지 여부를 TEST

알파 테스트

특정 사용자들에 의해 개발자 관점에서 수행되며, 개발자는 사용상의 문제를 기록하여 반영되도록 하는 TEST

베타 테스트

선정된 다수의 사용자들이 자신들의 사용환경에서 일정 기간 동안 사용해 보면서 문제점이나 개선 사항 등을 기록하고 개발 조직에게 통보하여 반영되도록 하는 TEST

◆ 설치 테스트

- 소프트웨어를 사용자 환경에 설치과정 중에 나타날 수 있는 결함을 발견할 목적으로 수행
- 하드웨어/소프트웨어 구성사항 테스트
- 파일분배 적재 테스트
- 타 소프트웨어와 연결관련 테스트

❖ 학습내용

[2] 테스트의 유형

1. 블랙박스 테스트

- ◆ 블랙박스 테스트의 정의
 - 블랙박스 시험(Black Box Test)은 원시코드는 보지 않은 채 목적코드를 수행시켜가 면서 결함을 발견할 수 있는 시험사례를 준비하여 시험에 임하는 방식으로 데이터 위주 또는 입출력 위주시험이라 함
 - 프로그램이나 정보시스템을 깊게 알지 못하는 순수한 사용자 관점에서 테스트
 - Dynamic Test 기법을 사용
- ◆ 블랙박스 테스트의 목적

- ✓ 부정확하거나 빠진 결함의 발견
- ✓ 인터페이스 결함 및 자료구조상의 결함 발견
- ✓ 성능 결함과 시작, 종결상의 결함 발견

◆ 블랙박스 테스트의 기법



- 다양한 입력 조건들을 갖춘 시험사례의 유형들로 분할
- 각 시험사례 유형마다 최소의 시험사례를 준비
- 시험 사례를 줄이기 위해 하나의 시험 사례가 비슷한 다른 유형의 시험 값에 대표될 수 있는 것으로 선정

❖ 학습내용

[2] 테스트의 유형

1. 블랙박스 테스트(계속)

◆ 블랙박스 테스트의 기법(계속)



- 입력조건 **의 경계치**에 치중하며 출력유형도 고려하여 시행
- **경계값을 기준**으로 경계값 내의 것, 경계값, 경계값 밖의 것으로 시험 사례 선정



- 입력 데이터 간의 관계가 **출력에 영향을 미치는 상황**을 체계적으로 분석하여 효율성 높은 시험사례를 발견하고자 하는 기법
- 인과 관계 그래프를 이용하여, 명세서의 **불완전성 및 애매모호함**을 추출



- 시험자의 **감각과 경험**으로 결함을 찾아보는 방식

❖ 학습내용

[2] 테스트의 유형

2. 화이트박스 테스트

- ◆ 화이트박스 테스트의 정의
 - 프로그램상에 허용되는 모든 논리적 경로를 파악하거나 경로들의 복잡도를 계산하여 시험사례를 만들어 시험을 수행하는 기법
 - 개발자의 관점에서 모든 소스를 하나하나 실행될 수 있도록 하여 테스트 함
 - Static Test기법을 사용
- ◆ 화이트박스 테스트에서의 시험영역

문장 영역	각 원시코드 라인이 한 번이라도 수행되도록 설계
물리적 경로영역	프로그램의 모든 경로가 한 번이라도 수행되도록 설계
논리적 경로영역	물리적 경로의 순서가 결과에 영향을 미친다는 가정 하에 논리적 경로들이 수행되도록 설계

- ◆ 화이트박스 테스트 조건
 - 프로그램 내의 소스들은 적어도 한번은 수행 되어야 함
 - 프로그램내의 모든 결정(Decision)이 각각 참, 거짓 값을 적어도 한번은 가져야 함

예

$A = 2 \text{ or } X > 1$ 일 경우 참을 만족하는 분기라면,
 $A = 2$ 의 조건 하에서는 X 값에 관계 없이 모든 값이 참이 됨

- 결정내의 모든 조건은 각각 참, 거짓 값을 적어도 한번은 가져야 함

예

$A > 1, A \neq 1$

- 또한 결정 자체도 참, 거짓을 적어도 한번은 가질 수 있어야 함
- 프로그램내의 모든 수행 가능한 경로는 모두 수행 되어야 함

❖ 학습내용

[2] 테스트의 유형

2. 화이트박스 테스트(계속)

◆ 화이트박스 테스트의 종류

구조시험

- 프로그램의 논리적 복잡도를 측정 후 이 척도에 따라 수행시킬 **기본 경로들의 집합** 정의
- 시험영역을 현실적으로 최대화 시켜주며 **독립경로 발견의 자동화**가 강점

루프(Loop)시험 기법

- 프로그램의 **루프구조에 국한**해서 실시하는 시험 기법
- 단순한 루프, 중첩 루프, 연결 루프, 비구조적 루프

◆ 블랙박스 테스트와 화이트박스 테스트

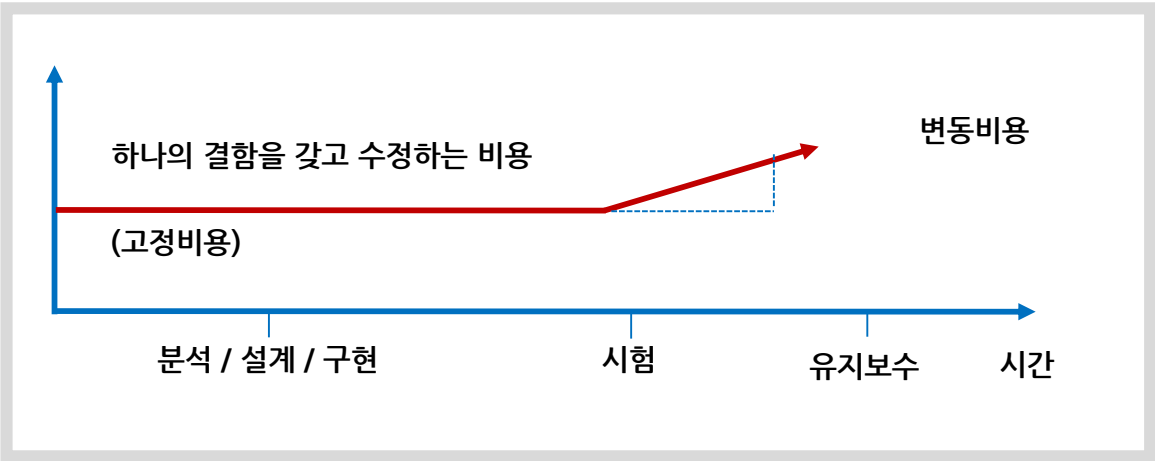
Black Box Test	White Box Test
<ul style="list-style-type: none">▪ 원시 코드는 보지 않은 채 목적 코드를 수행시켜 결함을 발견▪ 데이터 위주(Data-Driven) 혹은 입출력 위주(IO-Driven)▪ 대상 결함(부정확하거나 빠진 결함, 인터페이스 결함, 자료 구조상의 결함, 성능 결함, 시작과 종결상의 결함)	<ul style="list-style-type: none">▪ 논리적 경로를 파악하거나 경로의 복잡도를 이용▪ 시험 영역(문장 영역, 물리적 경로 영역, 논리적 경로)

❖ 학습내용

[2] 테스트의 유형

3. 테스트 고려사항

- ◆ 소프트웨어 테스트 유의사항
 - ① 테스트 모형 설계 시 예상출력 정의
 - ② 프로그래머는 자신의 프로그램 TEST 금지
 - ③ 테스트 결과의 철저한 분석의 요구
 - ④ 부당하거나 예기치 않은 입력도 테스트 등
 - ⑤ TEST CASE 유지(재사용) 등
 - ⑥ 시스템이 운영되는 시점에서 수정되는 비용보다 프로젝트 진행 초기 단계에 적절한 테스트를 통하여 프로그램이 수정되는 비용이 작음

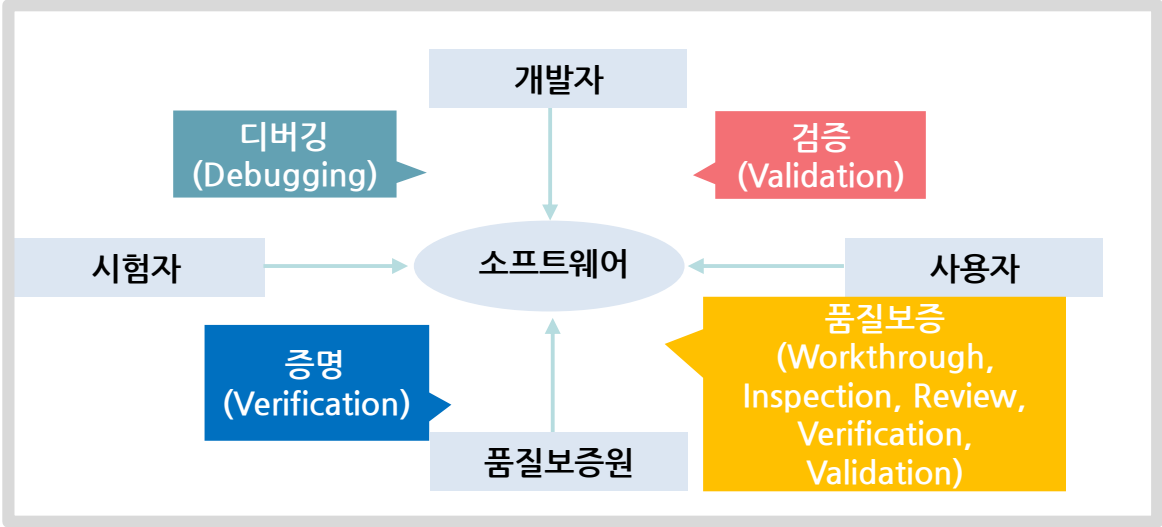


❖ 학습내용

[2] 테스트의 유형

3. 테스트 고려사항(계속)

◆ V&V, 테스트, 품질보증의 관계



◆ 효율적 테스트를 위한 고려 사항

- S/W 테스트는 **개발 및 시스템 구축의 완전성**을 위한 중요 요소
- **BLACK BOX** 및 **WHITE BOX** 테스트 방법을 개발 과정 중 보편적 사용
- 효율적인 S/W TEST 위해 **QA Check List Map**을 이용, 요구대비 테스트 목표달성 여부 분석, 최적 테스트 케이스/ 테스트 계획 준비
- 사용자가 참여하여 테스트 결과를 철저히 **검토, 기록, 수정**하고 **개선사항**을 반영
- 자동 테스트 케이스 생성 및 테스트 실시와 같은 **선진 기법** 필요
- 국가적으로 **테스트 툴**에 대한 연구개발 및 현장 도입이 필요

❖ 학습내용

[3] 소프트웨어 검사

1. 소프트웨어 품질

◆ 소프트웨어 품질 개요

소프트웨어 품질(Software Quality)

- 비즈니스 문맥에서 품질이 정의된 곳에 존재하는, 두 개의 서로 관련되면서도 구별된 개념을 의미
- 소프트웨어 품질은 소프트웨어 내부 구조, 소스 코드, 단위 수준, 기술 수준, 시스템 수준의 분석을 통해 평가

소프트웨어 기능상의 품질(Software Functional Quality)

- 기능 요건이나 사양에 기반하여 주어진 설계를 얼마나 잘 충족하고 있는지를 반영
- 소프트웨어의 목적이 부합하는지, 또 가치가 있는 상품으로서 시장의 경쟁작품과 견줄만한지를 기술

소프트웨어 구조상의 품질(Software Structural Quality)

- 기능 요건의 전달을 지원하는 비기능 요건을 어떻게 충족하는지를 의미
- 소프트웨어가 올바르게 개발될 수 있는지를 가늠하는 척도로서 내구성이나 유지보수성을 들 수 있음

❖ 학습내용

[3] 소프트웨어 검사

1. 소프트웨어 품질(계속)

- ◆ 운영단계에서 소프트웨어 품질
 - 정보시스템을 운영 시(즉 소프트웨어 사용 시) 다음과 같은 요소들이 소프트웨어의 품질을 좌우함

정확성 (Correctness)	▪ 사용자의 요구 정도 를 충족시키는 정도
신뢰성 (Reliability)	▪ 옳고 일관된 결과를 얻기 위해 요구된 기능 을 수행 할 수 있는 정도 → 사용자, 발주자, 유지보수자가 공통으로 관심을 보이는 항목
사용용이성 (Usability)	▪ 쉽게 사용할 수 있는 정도
유용성 (Utility)	▪ 사용자의 요구에 맞는 소프트웨어인 가를 평가하는 척도
무결성 (Integrity)	▪ 허용되지 않는 사용 이나 자료의 변경 을 허용 하지 않는 정도
효율성 (Efficiency)	▪ 최소의 시간 과 기억용량 을 소비하여 요구되는 기능을 수행할 수 있는 정도
- ◆ 정보시스템 개조, 수정단계에서 소프트웨어 품질
 - 기존 소프트웨어(정보시스템)을 수정, 개조 시 다음과 같은 요소가 소프트웨어의 품질을 좌우함

❖ 학습내용

[3] 소프트웨어 검사

1. 소프트웨어 품질(계속)

- ◆ 정보시스템 개조, 수정단계에서 소프트웨어 품질
 - 기존 소프트웨어(정보시스템)을 수정, 개조 시 다음과 같은 요소가 소프트웨어의 품질을 좌우함

유지보수성 (Maintainability)	오류가 발견되었을 때 쉽게 교정 되는 정도
유연성 (Flexibility)	기능의 추가나 다른 환경에서 적응하기 위해 쉽게 수정 될 수 있는 정도
검사용이성 (Testability)	쉽고 철저하게 테스트 될 수 있는 정도
이식성 (Portability)	여러 환경에서도 운용 가능하도록 쉽게 수정 될 수 있는 정도
재사용성 (Reusability)	전제나 일부가 다른 응용 목적 으로 사용될 수 있는 정도
상호운용성 (Interoperability)	다른 소프트웨어와 정보를 교환 할 수 있는 정도
강건성 (Robustness)	<ul style="list-style-type: none">■ 부적절한 입력 등에도 견뎌내는 정도■ 요구 명세서에 표시되지 않은 상황에서도 소프트웨어가 제대로 동작하는 성질

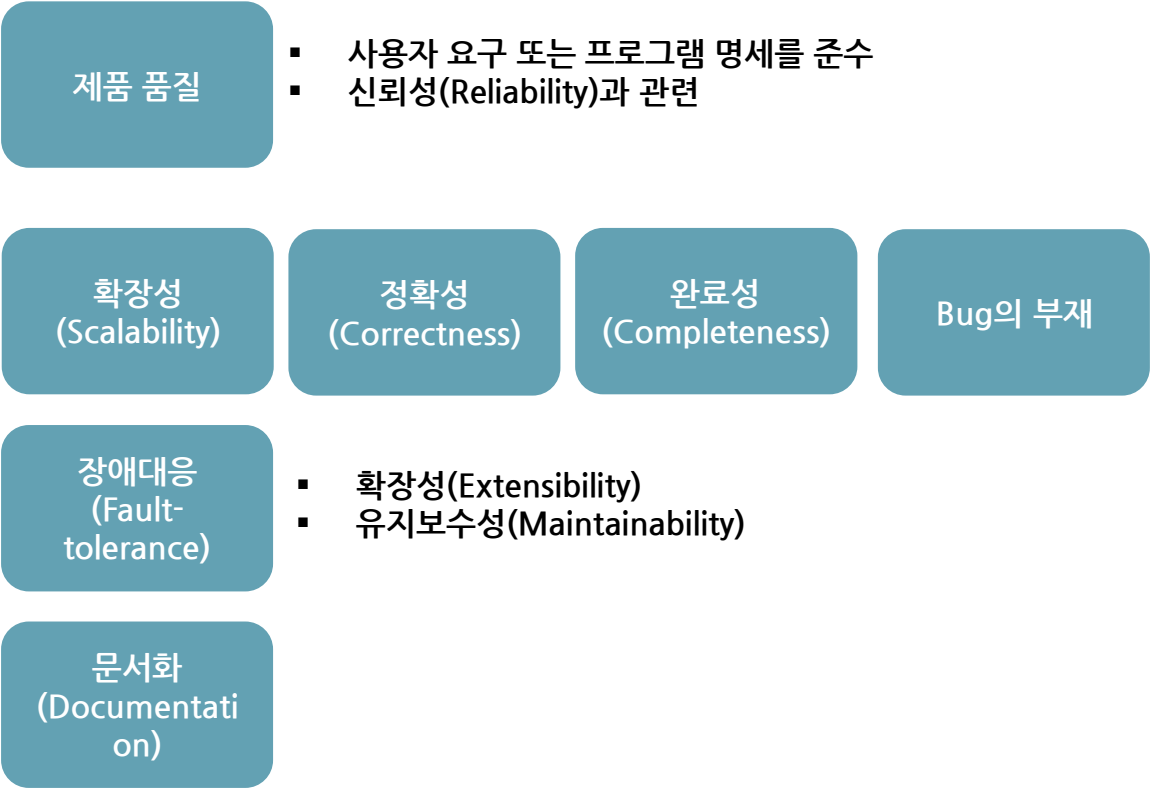
❖ 학습내용

[3] 소프트웨어 검사

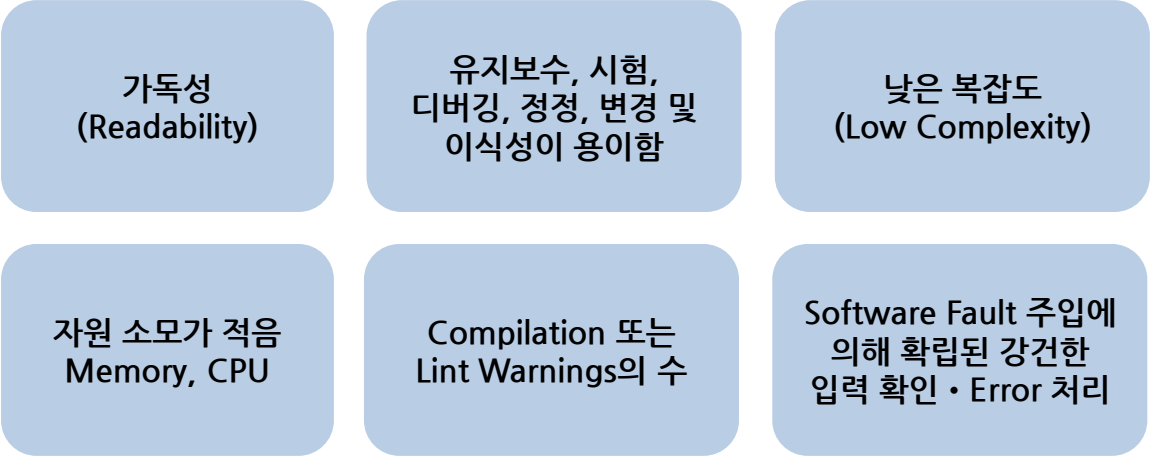
1. 소프트웨어 품질(계속)

◆ 소프트웨어 품질 측정 척도

▪ Software 제품 품질



▪ Source Code 품질



❖ 학습내용

[3] 소프트웨어 검사

1. 소프트웨어 품질(계속)

- ◆ 소프트웨어 검사 개요
 - 소프트웨어 시스템을 검토하여 오류, 생략, 이상을 찾기 위한 정적인 증명과 검증 프로세스
 - 검토회의(품질관리의 Inspection)를 더욱 발전시킨 소프트웨어 검사에서는 발견된 문제점들을 어떻게 수정해야 할지 그 지침까지도 제시
 - 또 개발자가 수정을 잘하고 있는지 추후에 조사도 함
 - 소프트웨어 개발 전 과정에 걸쳐 요구 분석 명세서, 설계 사양서, 원시 코드 뿐 아니라 각 단계 산출물의 문서 등을 포함하여 분석하고 품질을 평가
 - 개발자의 작업 결과가 표준 지침을 따르는지, 정해진 기준에 따라 수행하였는지 등을 개발자와 독립적으로 검사하여 조기에 작업 산출물의 결함을 효율적으로 제거, 같은 실수가 반복되지 않게 함
 - 소프트웨어 검사는 소프트웨어 품질 보증 기법의 하나임
- ◆ 소프트웨어 검사를 위한 고려사항
 - 소프트웨어 검사를 위하여 다음을 고려하여야 함

고려사항	내용
시험 프로세스	<ul style="list-style-type: none">▪ 앞 장에서 설명한 V-model을 고려하여야 함
요구사항 추적성	<ul style="list-style-type: none">▪ 사용자는 그들의 요구사항을 만족시키는 시스템에 관심을 가지고 모든 요구사항이 개별적으로 시험되도록 시험을 계획해야 함
소프트웨어 검사 항목	<ul style="list-style-type: none">▪ 소프트웨어 검사를 해야 할 소프트웨어 프로세스의 제품을 명세화해야 함
소프트웨어 검사 일정	<ul style="list-style-type: none">▪ 전체적인 시험 일정 및 자원 배정, 프로젝트의 개발일정들과 연관관계 파악
소프트웨어 검사 기록절차	<ul style="list-style-type: none">▪ 소프트웨어 검사를 위한 테스트를 수행하는 것만으로는 충분하지 않기 때문에, 이 테스트 결과를 체계적으로 기록하여야 함▪ 테스트 과정이 정확하게 수행되는지를 검토하기 위하여 시험과정을 감사(Audit)하는 것이 가능하여야 함

❖ 학습내용

[3] 소프트웨어 검사

1. 소프트웨어 품질(계속)

- ◆ 소프트웨어 검사를 위한 고려사항(계속)
 - 소프트웨어 검사를 위하여 다음을 고려하여야 함

고려사항	내용
하드웨어 및 소프트웨어 요구사항	<ul style="list-style-type: none">▪ 필요한 소프트웨어 도구 및 예측되는 추정된 하드웨어 이용을 기술하여야 함
계약조건	<ul style="list-style-type: none">▪ 인원의 부족을 미리 예측해야 하는 것과 같이 테스트 과정에 영향을 미치는 제약조건을 기술함

- ◆ 소프트웨어 검사 항목 개요
 - 다음 항목을 참고하여 소프트웨어 검사 수행

데이터 결함
<ul style="list-style-type: none">▪ 모든 프로그램 변수는 변수의 값이 사용되기 전에 선언되었는가?▪ 모든 상수는 이름이 있는가?▪ 배열의 상한은 배열의 크기인가 또는 배열의 크기보다 1이 작은가?▪ 만일 문자열이 사용되면, 구분자(Delimiter)가 명시적으로 지정되는가?▪ 버퍼 오버플로우의 가능성이 있는가?
제어 결함
<ul style="list-style-type: none">▪ 각 제어문에 대해서 조건이 정확한가?▪ 각 반복문은 확실히 종료되는가?▪ 복합문에 괄호가 적절하게 사용되는가?▪ Case문에 모든 가능한 경우가 다 고려되었는가?▪ Case문의 각 Case다음에 Break문이 요구된다면, 이것이 포함되었는가?

❖ 학습내용

[3] 소프트웨어 검사

1. 소프트웨어 품질(계속)

- ◆ 소프트웨어 검사 항목 개요(계속)
 - 다음 항목을 참고하여 소프트웨어 검사 수행(계속)

입출력 고장

- 모든 입력변수가 사용되는가?
- 모든 출력변수는 출력하기 전에 값이 지정되었는가?
- 예상치 못한 입력이 훼손을 야기할 수 있는가?

인터페이스 결함

- 모든 함수와 메소드 호출이 정확한 개수의 매개변수를 갖는가?
- 형식 매개변수와 실 매개변수의 형이 일치하는가?
- 매개변수들의 순서가 올바른가?
- 만일 컴포넌트들이 공유메모리에 접근하면, 공유메모리 구조에 대하여 동일한 모델을 갖는가?

기억장소 관리 결함

- 만일 연결된 구조가 수정되면 모든 링크들이 정확하게 재 할당되는가?
- 만일 동적 기억 장소가 사용되면, 기억장소를 정확하게 할당할 수 있는가?
- 기억장소가 더 이상 필요하지 않으면, 기억 장소를 명시적으로 해제하는가?

예외관리 결함

- 모든 가능한 오류조건을 고려하였는가?

❖ 핵심정리

1. 테스트의 단계

- 소프트웨어 테스트는 노출되지 않은 숨어있는 결함(Fault)을 찾기 위해 소프트웨어를 작동시키는 일련의 행위와 절차로 오류 발견을 목적으로 프로그램을 실행하여 품질을 평가하는 과정
- 단위 테스트는 설계의 최소 단위인 모듈을 TEST하는 과정
- 단위 테스트 이후 종합적으로 전체 소프트웨어를 TEST하는 종합테스트를 거치며, 이후 시스템테스트,인수 테스트를 수행함

2. 테스트의 유형

- 블랙박스 테스트는 원시코드는 보지 않은 채 목적코드를 수행시켜가면서 결함을 발견할 수 있는 시험사례를 준비하여 시험에 임하는 방식으로 데이터 위주 또는 입출력 위주시험이라 함
- 프로그램상에 허용되는 모든 논리적 경로를 파악하거나 경로들의 복잡도를 계산하여 시험사례를 만들어 시험을 수행하는 기법을 화이트 박스 테스트라 함

3. 소프트웨어 검사

- 검토회의(품질관리의 Inspection)를 더욱 발전시킨 소프트웨어 검사에서는 발견된 문제점들을 어떻게 수정해야 할지 그 지침까지도 제시
- 또 개발자가 수정을 잘하고 있는지 추후에 조사도 함