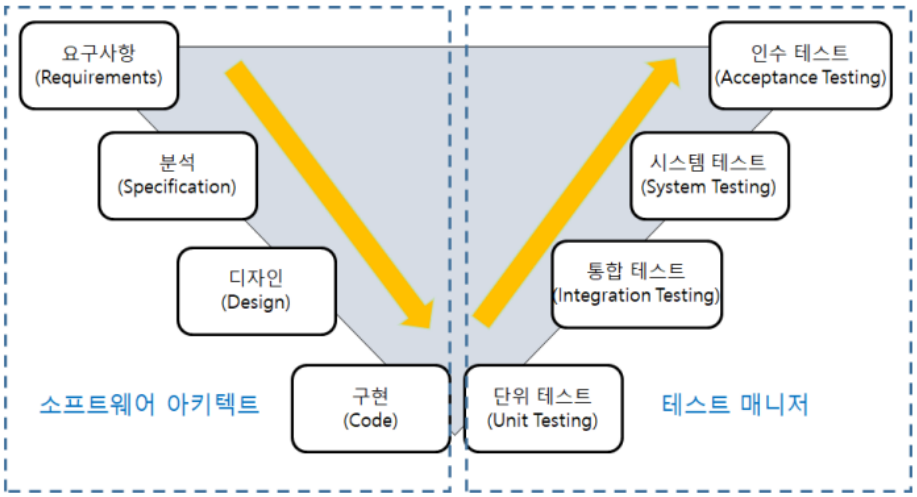
애플리케이션 테스트 수행

1. 애플리케이션 테스트 수행

1 - 1 테스트 수행

① 테스트의 개요

테스트 과정에 필요한 역할은 소프트웨어 아키텍트와 테스트 매니저이다. 두 역할은 소프트웨어 생명 주기(Life Cycle)의 V 모델에서 각각 좌측과 우측의 핵심 역할을 담당하고 서로 보완 관계에 있다. 소프트웨어 생명 주기는 요구사항, 분석, 디자인, 구현 또는 개발 순으로 진행되며, 프로젝트의 특성과 방법론에 따라 반복적(iteration)으로 수행하는 경우도 있다. 테스트는 단위 테스트, 통합 테스트, 시스템 테스트, 인수 테스트의 순으로 진행된다



[그림 1-1] 소프트웨어 생명 주기의 V-모델

1. 프로젝트 수행 단계에 따른 테스트의 분류

(1) 단위 테스트

작은 소프트웨어 단위(컴포넌트 또는 모듈)를 테스트하는 것으로서, 일반적으로 개발자 자신에 의해 행해진다. 과거에는 시간 부족을 이유로 단위 테스트가 생략되었으나 최근에는 개발 도구의 발전으로 개발 과정 중에 자동으로 진행된다. 단위 테스트는 아주 중요한 부분이므로 개발 도구에서 지원하지 않아도 반드시 수행해야 한다.

(2) 통합 테스트

모듈 사이의 인터페이스, 통합된 컴포넌트 간의 상호 작용을 테스트하는 것으로, 하나의 프로세스가 완성된 경우 부분적으로 통합 테스트를 수행하는 경우도 있다

(3) 시스템 테스트

통합된 단위 시스템의 기능이 시스템에서 정상적으로 수행되는지를 테스트하는 것으로 성능 및 장애 테스트가 여기에 포함된다

(4) 인수 테스트

일반적으로 최종 사용자와 업무에 따른 이해관계자 등이 테스트를 수행함으로써 개발된 제품에 대해 운영 여부를 결정하는 테스트로, 실제 업무 적용 전에 수행한다

② 프로젝트 수행 단계에 따른 테스트의 접근 방법

1. 단위 테스트

(1) 특징

구조적 테스트, 기능성 테스트, 리소스 관련 테스트, 강건성 테스트 등 특정 비기능성 테스트 등이 포함되어 수행되며, 컴포넌트 명세, 소프트웨어 상세설계, 데이터 모델 명세 등을 이용하여 테스트한다.

(2) 방법과 목적

(가) 구조 기반은 업무 단위별 제어 흐름과 조건 결정에 따른 결과를 테스트하는 데 목적이 있다

(나) 명세 기반은 동등 분할과 경계 값 분석을 위하여 사용자의 입력, 출력, 내부, 이벤트 등을 확인하는 데 목적이 있다

〈표 1-1〉 단위 테스트 방법

테스트 방법	테스트 내용	테스트 목적
구조 기반	- 프로그램 내부 구조 및 복잡도를 검증하는 화이트박스(White Box) 테스트	제어 흐름, 조건 결정
명세 기반	- 목적 및 실행 코드 기반의 실행을 통한 블랙박스(Black Box) 테스트	동등 분할, 경계 값 분석

2. 통합 테스트

(1) 특징

일반적으로 빅뱅 방식보다는 순차적(Incremental) 형태와 아키텍처에 대한 이해를 바탕으로 진행한다.

(2) 종류

빅뱅, 상향, 하향, 샌드위치, Central, Collaboration, 레이어 통합 등의 테스트 가 있다

3. 시스템 테스트

(1) 특징

시스템 테스트는 개발 프로젝트 차원에서 정의된 전체 시스템의 동작과 관련되어 있다. 환경 제한적 장애 관련 리스크를 최소화하기 위하여 실제의 최종 사용자 환경과 유사하게 시스템 성능, 관련된 고객의 기능, 비기능적인 요구사항 등이 완벽하게 수행되는지를 테스트하며, 이때 요구사항 명세서, 비즈니스 절차, 유스케이스, 리스크 분석 결과 등을 이용한다.

(2) 테스트 방법

〈표 1-2〉 통합 테스트 방법

테스트 방법	테스트 내용
기능적 요구사항	요구사항 명세서, 비즈니스 절차, 유스케이스 등 명세서 기반의 블랙박스(Black Box) 테스트
비기능적 요구사항	성능 테스트, 회복 테스트, 보안 테스트, 내부 시스템의 메뉴 구조, 웹 페이지의 네비게이션 등의 구조적 요소에 대한 화이트박스(White Box) 테스트

4. 인수 테스트

〈표 1-3〉 인수 테스트 종류와 내용

테스트 종류	테스트 내용
사용자 인수 테스트	비즈니스 사용자가 시스템 사용의 적절성 여부를 확인
운영상의 인수 테스트	시스템 관리자가 시스템 인수 시 수행하는 테스트 활동으로 백업/복원 시스템, 재난 복구, 사용자 관리, 정기 점검 등을 확인
계약 인수 테스트	계약상의 인수/검수 조건을 준수하는지 여부를 확인
규정 인수 테스트	정부 지침, 법규, 규정 등 규정에 맞게 개발하였는지 확인
알파 테스트	개발하는 조직 내 잠재 고객에 의해 테스트 수행
베타 테스트	실제 환경에서 고객에 의해 테스트 수행

④ 테스트 자동화 도구

2. 테스트 자동화

(1) 테스트 자동화의 개념

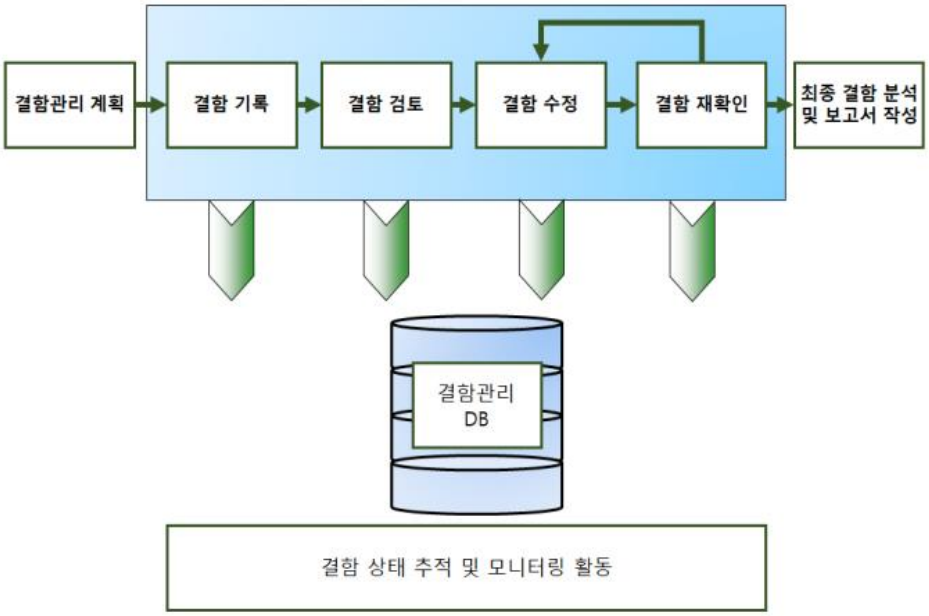
테스트 자동화란 사람이 하던 반복적 테스트 절차를 자동화 도구를 활용하여 준비, 구현, 수행, 분석 등을 스크립트 형태로 구현함으로써, 시간과 인력 투입의 부담을 최소화하면서 운영 중인 시스템의 모니터링 또는 UI가 없는 서비스의 경우에도 정밀한 테스트가 가능하도록 하는 것이다.

〈표 1-5〉 테스트 활동에 따른 자동화 도구

테스트 활동	테스트 도구	내 용
테스트 계획	요구사항 관리	고객 요구사항 정의 및 변경 사항 관리
테스트 분석/설계	테스트케이스 생성	테스트 기법에 따른 테스트 데이터 및 케이스 작성
	커버리지 분석	대상 시스템에 대한 테스트 완료 범위의 척도
테스트 수행	테스트 자동화	기능 테스트 등 테스트 도구를 활용하여 자동화를 통한 테스트의 효율성 제고
	정적 분석	코딩 표준, 런타임 오류 등을 검증
	동적 분석	대상 시스템 시뮬레이션을 통한 오류 검출
	성능 테스트	가상 사용자를 인위적으로 생성하여 시스템 처리능력 측정
	모니터링	시스템 자원(CPU, Memory 등)의 상태 확인 및 분석 지원 도구
테스트 통제	형상 관리	테스트 수행에 필요한 다양한 도구 및 데이터 관리
	테스트 관리	전반적인 테스트 계획 및 활동에 대한 관리
	결함 추적/관리	테스트에서 발생한 결함 관리 및 협업 지원

출처: 공개 SW 포털 홈페이지. 공개SW테스트(http://www.oss.kr/oss_repository6/69515)에서 2017. 8. 21, 검색.

② 결함관리 프로세스



[그림 1-6] 결함관리 프로세스 흐름도

1. **결함 관리 계획**

   결함관리 계획은 전체 프로세스에서 결함관리에 대한 일정, 인력, 업무 프로세스를 확보하여 계획을 수립하는 것을 말한다.

2. **결함 기록**

   테스터는 발견된 결함에 대한 정보를 결함관리 DB에 기록한다.

3. **결함 검토**

   등록된 결함에 있어서 주요 내용을 검토하고, 결함을 수정할 개발자에게 전달한다.

4. **결함 수정**

   개발자는 할당된 결함의 프로그램을 수정한다.

5. **결함 재확인**

   테스터는 개발자가 수정한 내용을 확인하고 다시 테스트를 수행한다.

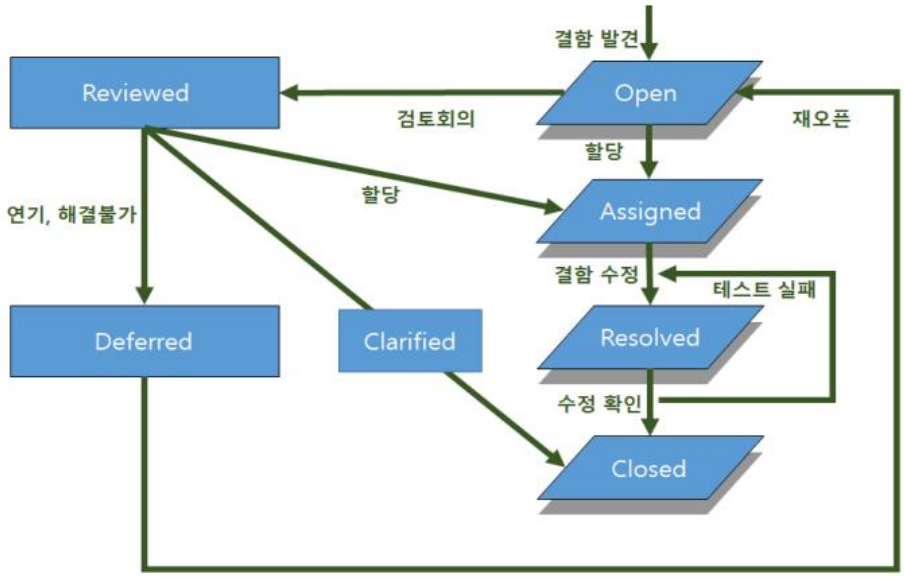
6. **결함 상태 추적 및 모니터링 활동**

   결함관리 팀장은 결함관리 데이터베이스를 이용하여 대시보드, 게시판 형태의 서비스를 제공한다.

7. **최종 결함 분석 및 보고서 작성**

   발견된 결함에 대한 내용과 이해관계자들의 의견이 반영된 보고서를 작성하고 결함 관리를 종료한다.

③ 결함의 상태 및 추적



[그림 1-7] 결함의 상태 및 추적 업무 흐름도

1. **결함 등록(Open)**

   테스터와 품질 관리(QA) 담당자에 의해 결함이 처음 발견되어 등록되었지만,
   아직 분석이 되지 않은 상태이다.

2. **결함 검토(Reviewed)**

   등록된 결함을 담당 모듈 개발자, 테스터, 프로그램 리더, 품질 관리(QA) 담당자와 검토하는 상태.

3. **결함 할당(Assigned)**

   결함의 영향 분석 및 수정을 위해 개발자와 문제 해결 담당자에게 할당된 상태이다.

4. **결함 수정(Resolved)**

   개발자에 의해 결함의 수정이 완료된 상태이다.

5. **결함 조치 보류(Deferred)**

   수정이 필요한 결함이지만 현재 수정이 불가능해서 연기된 상태로서 우선순위, 일정 등을
   고려하여 재오픈을 준비하는 상태이다

6. **결함 종료(Closed)**

   발견된 결함이 해결되고 테스터와 품질 관리(QA) 담당자에 의해 종료 승인을 한 상태이다

7. **결함 해제(Clarified)**

   테스터, 프로그램 리더, 품질 관리(QA) 담당자가 결함을 검토한 결과, 결함이 아니라고 판명된 경우.

④ 결함 분류

1. 시스템 결함

비정상적인 종료/중단, 응답 시간 지연, 데이터베이스 에러 등 주로 애플리케이션 환경과 데이터베이스 처리에서 발생하는 결함을 말한다.

(1) 비정상적인 종료/중단

특정 기능 실행 시 응용 프로그램의 작동 정지, 종료, 시스템 다운이 되는 경우이다.

(2) 응답 시간 지연

응용 프로그램 작동 후 조회 또는 보고서 출력 시 지연되는 경우와 메모리 부족, 하드웨어와 소프트웨어의 비 일관성으로 발생되는 경우이다.

(3) 데이터베이스 에러

응용 프로그램 작동 후 사용자 데이터의 등록, 수정, 삭제, 조회가 정상적으로 작동하지 않는 경우.

2. 기능 결함

사용자의 요구사항 미반영/불일치, 부정확한 비즈니스 프로세스, 스크립트 에러, 타 시스템 연동 시 오류 등 기획, 설계, 업무 시나리오 단계에서 발생된 결함을 말한다.

(1) 요구사항 미반영/불일치

요구사항에 명시된 기능이 응용 프로그램에 구현되지 않은 경우와, 다르게 구현되어진 경우이다.

(2) 부정확한 비즈니스 프로세스

기능 자체는 수행되나 내부 프로세스 로직의 문제로 부정확한 결과를 내는 경우이다.

(3) 스크립트 에러

특정 기능 실행 시 웹 브라우저에서 스크립트 오류가 발생하는 경우이다.

(4) 타 시스템 연동 시 오류

기존 시스템과의 연동을 통해 데이터를 주고받는 과정에서 오류가 발생하는 경우이다.

3. GUI 결함

GUI 결함은 응용 프로그램의 UI 비일관성, 부정확한 커서/메시지, 데이터 타입의 표시 오류 등으로 사용자 화면 설계에서 발생된 결함을 말한다.

(1) 응용 프로그램 UI 비일관성

프로젝트에서 정의한 UI 표준과 상이하게 구현된 경우이다.

(2) 부정확한 커서/메시지

커서의 위치가 입력 대상의 첫 번째 필드에 위치해 있지 않거나, 탭 시퀀스가 순차적으로 동작하지 않는 경우, 각 기능에서 제공하는 메시지 내용이 부정확한 내용을 보여주는 경우이다.

(3) 데이터 타입의 표시 오류

입력 필드에 지정된 형식과 다르게 입력해도 저장이 되는 경우와 입력 필드에 유효하지 않은 데이터(Invalid Data)를 입력했을 때 오류가 나는 경우이다.

4. 문서 결함

기획자, 사용자, 개발자 간의 의사소통과 기록이 원활하지 않은 경우에 발생하는 결함으로 사용자의 온라인/오프라인 매뉴얼의 불일치, 요구사항 분석서와 기능 요구사항의 불일치로 인한 불완전한 상태의 문서의 경우를 말한다.

2 - 1 조치 우선순위 결정

① 소프트웨어 테스트 기법

1. 단위 테스트 기법

(1) JUnit을 활용한 테스트

Java환경 시 대부분 JUnit이라는 단위 테스트 프레임워크를 통해 단위 테스트를 할 수 있어야 한다.

〈표 2-1〉 테스트 클래스 작성 방법

단계	수행 내용
1	junit.framework.TestCase의 서브 클래스 타입의 테스터 클래스를 생성한다.
2	생성자를 작성한다.
3	setUp, tearDown 메소드를 오버라이드한다.
4	일반적으로 Junit 테스트에서는 Fixture의 메소드를 테스트하게 되는데, 이때 Fixture의 테스트할 메소드에 매핑이 되는 메소드를 작성한다.
5	TestRunner 클래스의 static 메소드인 run 메소드를 호출한다.

출처 : 표준프레임워크 홈페이지 UNIT TEST
(http://www.egovframe.go.kr/wiki/doku.php?id=egovframework:dev2:tst:)에서 2017. 08. 21. 검색.

(2) Mock 테스트

(가) 단위 테스트 시 Mock 객체를 사용하여 테스트하는 기법을 말하며, 특정 기능 또는 모듈에 대한 응답 결과를 미리 정의해 놓고 테스트한다. 이는 특정 모듈이나 기능이 완벽히 개발 완료되지 않은 상태에서도 진행할 수 있다.

(나) 테스트 전용 객체를 테스트 더블(Test Double)이라 부르며, 이는 테스트를 위해 실제 객체를 대신해서 사용되는 용어이고 객체의 유형은 와 같다

〈표 2-2〉 Mock 테스트 객체 유형

객체 유형	수행 내용
Dummy	객체의 전달에만 사용되고 실제로는 사용하지 않는 것이다. 주로 매개변수 목록을 채우는 데 쓰인다.
Fake	실제로 동작하도록 구현되지만, 보통 빠른 구현을 위해 실제 환경과는 다르게 구현할 수 있다.
Stubs	테스트를 위해 미리 준비한 응답만을 제공하는 것이다. 그 외의 상황에 대해서는 정상적인 작동을 하지 못하는 것이 일반적이다. 또한 스텁은 호출에 대한 정보를 기록하는 경우도 있다.
Mocks	스펙을 통해 정의된 응답을 받고 다른 응답을 받을 경우 예외를 발생하도록 구현되어 있으며, 응답에 대한 확인을 수행하는 역할을 한다.

출처 : 표준프레임워크 홈페이지 Mock
Support(http://www.egovframe.go.kr/wiki/doku.php?id=egovframework:dev2:tst:mock_support)에서 2017.08.21. 검색

2. 통합 테스트 기법

전체 시스템이 통합 완료될 때까지 단위 시스템 간의 연계성 및 기능 요구사항들을 확인하고, 하드웨어와 소프트웨어 구성 요소 간의 상호 작용을 테스트하는 것이 주요 목적이다. 업무 간의 연계성과 상호 운영성 중심의 테스트를 수행한다

(1) 테스트 설계 기법

테스트 설계는 개발된 소프트웨어나 시스템의 요구사항, 요구사항 명세서, 업무 구조, 시스템 구조 등을 기반으로 소프트웨어의 어떤 부분을 어떻게 접근하여 테스트할지에 대한 테스트 상황과 방법을 파악하는 것이다. 이를 체계적으로 구체화시켜 테스트 케이스를 도출하고 작성하는 것을 테스트 구현이라고 하고, 테스트 상황과 방법을 구체화시키기 위한 수단 및 도구를 테스트 설계 방법이라고 한다

(2) 테스트 설계 방법

분류	수행 내용
보다 작은 케이스	동등 클래스(Equivalence Class) 예) 한 자리 정수를 더하는 프로그램인 경우 1+1, 1+2…9+9까지 81가지 케이스는 모두 동등한 경우이다. 따라서 이런 경우를 모두 모아서 몇 가지 케이스만 테스트하면 되는 경우이다.
보다 많은 버그 찾기	경계 테스트(Boundary Test) 예) 위의 동등 클래스 중 대표 클래스를 뽑을 때 가장자리, 즉 경계값을 뽑는 경우이다.
테스트 대상을 빠짐없이 실행하기	모델 기초(Model-based) 테스트 설계 예) 상태 전이(State-transition) 테스트 모델을 UML의 State Chart Diagram으로 표현하여 노드와 링크를 모두 대상으로 하여 테스트 한다. 또는 모든 테스트 요소를 매트릭스 형태로 나열하여 테스트 한다.

3. 시스템 테스트 기법

(1) 부하 및 성능 테스트

〈표 2-4〉 부하 테스트 기본 공식

분류	기본 공식 내용
동시 이용자 수 (Concurrent User)	TPS(Throughput)*호출간격(응답시간(Sec)+대기시간(Sec))
동시 단말 이용자 (Concurrent User)	PC 앞에 앉아서 시스템을 이용하는 이용자로서, Active User와 (가상적으로) In-active User의 합으로 정의한다.
액티브 이용자 (Active User)	동시에 같은 서비스나 업무를 실행하고 나서 응답을 기다리고 있는 이용자
처리량 (Throughtput)	단위 시간당 처리하는 건수. 단위는 tps(transaction per second), tpm, tph를 사용하고, 관점에 따라 두 가지 유형이 있으며, 요청을 받는 입장에서 단위 시간당 요청 건수(Arrival rate)와 단위 시간당 처리 건수(Service Rate)로 구분되어 표현한다.
대기 시간 (Thinktime)	응답을 받은 직후부터 다음 명령 또는 호출할 때까지 사용자가 대기하는 시간

(2) 장애 복구 테스트

<표 2-5> 장애 복구 테스트 예시

테스트 유형	테스트 시나리오	측정 결과 기준
웹 서버 장애	- 동작 중인 웹 서버 10대 중 1대의 웹 서버 프로세스를 강제로 Kill한다.	서비스 지속적 유지
웹 애플리케이션 장애	- 동작 중인 WAS서버의 인스턴스 8개 중 실행 중인 인스턴스 한 개를 멈춘다. - 현재 접속 중인 사용자의 수행 인스턴스여야 한다.	자동 세션 및, 서비스 유지
데이터베이스 장애	- HA(High Availability) Fail-over, Fail-back 시 rerouting 동작 여부를 테스트한다. - Active한 Primary 서버의 DB서비스를 강제로 Kill한다.	10초 내 서비스 복구

(3) 보안 테스트

보안 테스트는 별도의 보안 전문가에 의해 애플리케이션 전반에 걸쳐 검증받는 것이 바람직하다. 보안성 검증은 애플리케이션의 기능적 요구사항과는 별도로 해 킹이나 침투에 대비할 수 있는 최신 기술 트렌드가 반영된 검증이 필요하며, 보안 검증이 완료되면 결과를 바탕으로 취약점 분석을 한 후 개선 방안을 수립하고 반드시 이를 시스템에 반영하도록 계획을 수립해야 한다.

4. 인수 테스트 기법

최종 사용자가 요구한 기능이 제대로 반영되었는지, 인수 조건에 만족하는지를 테스트하는 기법이다. 요구 기능 만족 여부, 사용 편리성에 대하여 실제 운영 환경에서 실행되며 고객이 주도하는 테스트.

② 결함관리의 이해

1. 결함 관련 용어

(1) 에러(Error)

소프트웨어 개발 또는 유지 보수 수행 중에 발생한 부정확한 결과로, 개발자의 실수로 발생한 오타, 개발 명세서의 잘못된 이해, 서브루틴의 기능 오해 등이 있다.

(2) 오류(Fault)

프로그램 코드 상에 존재하는 것으로 비정상적인 프로그램과 정상적인 프로그램 버전 간의 차이로 인하여 발생되며, 잘못된 연산자가 사용된 경우에 프로그램이 서브루틴으로부터의 에러 리턴을 점검하는 코드가 누락된 것을 말한다.

(3) 실패(Failure)

정상적인 프로그램과 비정상적인 프로그램의 실행 결과의 차이를 의미하며, 프로그램 실행 중에

프로그램의 실제 실행 결과를 개발 명세서에 정의된 예상 결과와 비교함으로써 발견한다.

(4) 결함(Defect)

버그, 에러, 오류, 실패, 프로그램 실행에 대한 문제점, 프로그램 개선 사항 등 전체를 포괄하는 용어.

2. 결함의 판단 기준

(1) 기능 명세서에 가능하다고 명시된 동작을 수행하지 않는 경우
(2) 기능 명세서에 불가능하다고 명시된 동작을 수행하는 경우
(3) 기능 명세서에 명시되어 있지 않은 동작을 수행하는 경우
(4) 기능 명세서에 명시되어 있지 않지만 수행해야 할 동작을 수행하지 않는 경우
(5) 테스터의 시각에서 볼 때 문제가 있다고 판단되는 경우

2 - 2 결함 조치 관리

① 프로그램 코드 검토 기법

3. 코드 인스펙션의 프로세스와 수행 내용

(1) 코드 인스펙션 프로세스

〈표 2-14〉 코드 인스펙션 프로세스

구분	수행 단계	주요 내용
자동 수행	1. 범위 계획 (Capacity Plan)	얼마만큼 인스펙션할 것인가? 인스펙션의 범위와 범위 선정 기준을 결정함.
	2. 시작 (Overview)	자동 인스펙션 수행
준비 단계	3. 준비 (Preparation)	계획서 작성, 체크리스트 작성, 계획 공지, 대상 산출물 준비
이행 단계	4. 인스펙션 회의 (Inspection Meeting)	사전 검토 실시, 미팅 실시
시정 조치	5. 재작업 (Rework)	개발 원작자가 직접 작업함.
	6. 후속처리 (Follow-up)	결과 분석서 작성 및 보고

(2) 코드 인스펙션 태스크별 수행 내용

〈표 2-15〉 코드 인스펙션 태스크와 수행 내용

구분	태스크	수행 내용	산출물
자동 수행	1. 자동 인스펙션 수행	- 전수 검사, Quality Metric, 결함 분석	코드 인스펙션 결과
준비 단계	2. 계획서 작성	- 일정 및 관련자, 대상 산출물 및 준비물 정의	인스펙션 계획서
	3. 체크리스트 작성	- 표준 체크리스트를 테일러링하여 인스펙션 체크리스트 작성	인스펙션 체크리스트
	4. 계획 공지	- 메일이나 공지를 통해 관련자에게 사전 공지	
	5. 대상 산출물 준비	- 산출물 작성자가 인스펙션 시 필요한 자료를 준비	
이행 단계	6. 착수 회의 실시	- 진행자는 참여자에게 검토 주안점, 검토 방법, 역할 등을 교육 - 작성자는 대상 산출물 및 참조 자료에 대한 개요 소개	
	7. 사전 검토 실시	- 산출물, 체크리스트, 사전검토서 양식 배포 - 참여자별로 자료를 개별 검토하여 발견된 부적합 사항 기록	인스펙션 결과서
	8. 미팅 실시	- 사전 검토에서 발견한 부적합 사항 검증 - 미팅에서 발견된 부적합 사항 추가 기재 - 부적합 사항 목록 정리 또는 조치 계획 수립	인스펙션 결과서
	9. 결과 정리	- 진행자는 최종 확정된 결함 내용을 인스펙션 결과서에 정리한 후 작성자에게 배포	
시정 조치	10. 보완 작업 실시	- 작성자는 각 결함에 대한 보완 작업을 실시	
	11. 시정 조치 결과 확인	- 진행자는 보완 완료 여부를 확인 (필요 시 재검토 실시)	
	12. 결과 보고	- 진행자는 결과서를 작성하여 관련자에게 보고	인스펙션 결과서

② 형상 관리 및 구성 요소

1. 소프트웨어 형상 관리의 정의

(1) 소프트웨어 프로세스의 모든 출력물 정보

(2) 컴퓨터 프로그램, 컴퓨터 프로그램 설명 문서, 데이터 등

(3) 소프트웨어 프로세스 전반에 걸쳐 소프트웨어 형상의 변경 요인에 대해 소프트웨어 형상을 보호하는 활동

〈표 2-17〉 소프트웨어 형상 관리와 소프트웨어 지원 비교 예시

소프트웨어 형상 관리	소프트웨어 지원
소프트웨어 엔지니어링 프로젝트 개시에서 소프트웨어 소멸 시점까지의 활동	소프트웨어가 고객에게 인도되고 운영되는 시점에 발생하는 소프트웨어 엔지니어링 활동

2. 기준선과 소프트웨어 형상 항목

(1) 기준선(Baseline)

변경을 통제하게 도와주는 기준선은 정식으로 검토 및 합의된 명세서나 제품 개발의 바탕으로서 정식의 변경 통제 절차를 통해서만 변경 가능하다

(2) 소프트웨어 형상 항목(SCI. Software Configuration Item)

소프트웨어 형상과 개발 도구의 합성으로서, SCI는 과 같이 개발 단계별로 기준선을 기준으로 형상 항목을 관리한다

〈표 2-18〉 소프트웨어 형상 항목 예시

개발 단계	기준선(Baseline)	소프트웨어 형상 항목
계획	사용자 요구사항	시스템 명세서, 개발계획서, 구성관리계획서, 품질평가계획서, 개발표준 및 절차 매뉴얼
요구 분석	사용자 요구 기능이 하위 시스템 간에 어떻게 분배되는가 여부	자료흐름도, 자료사전, 자료흐름도 명세서
설계	개발 전 설계 명세	입출력명세서, 화면설계서, 초기 사용자 매뉴얼, 초기 시스템 매뉴얼, 자료 구조도, 시스템 구조도
구현	시험 계획서	원시코드, 목적코드, 실행코드, 단위시험 보고서
시스템 통합 및 시험	제품	통합시험 보고서, 기능/성능/과부하 시험 보고서, 인증시험 보고서
설치 및 운영	운영	목적/실행코드, 운영자 매뉴얼, 사용자 매뉴얼