**화이트 박스 테스트**

* **화이트 박스 테스트**는 소프트웨어의 내부 구조와 설계가 어떻게 작동하는지를 테스트하는 방법
* 테스트는 **블랙 박스 테스트**와 대조되며, 블랙 박스 테스트는 소프트웨어의 내부 작동이 아닌 외부 출력만을 테스트.
* 화이트 박스 테스트는 다음과 같은 다양한 소프트웨어 테스트 유형을 포함.
  1. **단위 테스트 (Unit Testing)**:

개별 코드 단위를 검증하는 테스트.

* 1. **통합 테스트 (Integration Testing)**:

여러 모듈이 함께 작동하는지를 검증하는 테스트.

* 화이트 박스 테스트는 **코드와 프로그래밍에 대한 이해**가 필요하며,

소프트웨어 엔지니어링에서 중요한 단계.

**화이트 박스 테스트의 필요성**

* 화이트 박스 테스트는 소프트웨어 테스트 및 엔지니어링에서 **내부 코드와 구조의**

**기능을 검증**하기 위해 다양한 단계에서 수행.

* 테스트는 다음과 같은 상황에서 필요.
  1. **단위 테스트**:

개발자가 개별 단위를 검증할 때.

* 1. **통합 테스트**:

여러 모듈이 함께 작동하는지를 확인할 때.

* 화이트 박스 테스트는 **테스트 범위를 극대화**하고, 특정 코드 섹션의 작동 방식을

검증하는 데 유용

* **보안 결함** 및 기타 취약점을 식별하는 데에도 효과적

**화이트 박스 테스트의 참여자**

* 화이트 박스 테스트는 거의 항상 **소프트웨어 개발자와 엔지니어**에 의해 수행
* 테스트는 다음과 같은 이유로 QA 테스터가 수행하기 어려움
  1. **상세한 코드 지식**:

화이트 박스 테스트는 코드의 내부 작동을 이해해야 하므로, 프로그래밍

지식이 필수적.

* 1. **단위 테스트**:

주로 개발 환경에서 수행되며, 개발자가 직접 검증.

* 개발자는 필요에 따라 화이트 박스 테스트를 수행하여 **버그 수정** 여부를 확인할

수 있음.

**화이트 박스 테스트의 장점**

* 화이트 박스 테스트는 개발자와 소프트웨어 엔지니어가 **코드의 더 많은 측면을**

**테스트**할 수 있도록 함.

* 주요 장점:
  1. **테스트 범위 극대화**:

가능한 많은 코드가 테스트되어 버그를 발견할 확률이 높아짐.

* 1. **숨겨진 오류 및 버그 발견**:

내부 기능을 검증함으로써 코드 깊숙이 숨겨진 오류를 찾기 용이

* 1. **자동화 용이성**:

단위 테스트는 자동화하기 쉬워, 빠르고 효율적인 테스트가 가능

* 1. **시간 효율성**:

자동화 덕분에 화이트 박스 테스트는 블랙 박스 테스트보다

빠르게 수행될 수 있음.

**화이트 박스 테스트의 도전 과제**

* 화이트 박스 테스트는 블랙 박스 테스트보다 수행하기 더 어려운 몇 가지 이유가

있음.

* 주요 도전 과제:
  1. **기술적 장벽**:

내부 시스템의 작동을 이해해야 하므로 프로그래밍 지식이 필요

* 1. **비용**:

화이트 박스 테스트는 철저하게 수행해야 하므로 비용이 더 많이 듬.

* 1. **정확성**:

화이트 박스 테스트는 이미 구현된 기능만 검증하므로, 누락된 기능을

발견하기 어려움.

* 1. **범위 제한**: 사용자 경험이나 최종 결과에 대한 정보는 제공하지 않으므로,

블랙 박스 테스트와 결합.

**화이트 박스 테스트의 특성**

* 화이트 박스 테스트는 블랙 박스 테스트와 구별되는 특정 특성으로 정의될 수 있음.
* 주요 특성:
  1. **유지 관리 용이성**:

코드의 내부 작동을 지속적으로 검토함으로써 유지 관리가 용이.

* 1. **유연성**:

코드가 변경될 수 있는 유연성을 제공하여 문제를 조기에 해결.

* 1. **모듈성**:

코드의 각 요소가 명확히 구분되어 있어야 테스트가 용이.

* 1. **통합 테스트에 유용**:

기능이 소프트웨어를 떠나기 전까지 작동하는지 확인.

**화이트 박스 테스트에서 검증하는 것**

* 화이트 박스 테스트는 블랙 박스 테스트 방법으로 검증할 수 없는 코드의 기능을

테스트하는 데 사용.

* 검증하는 주요 요소.
  1. **내부 보안 구멍**:

해커가 악용할 수 있는 보안 취약점을 찾음.

* 1. **코딩 프로세스의 경로**:

코드의 논리와 구조를 검증.

* 1. **예상 출력**:

코드의 입력에 대한 예상 출력을 검증.

* 1. **조건부 루프의 기능**:

루프가 효율적으로 작동하는지 확인.

**화이트 박스 테스트의 종류**

* 화이트 박스 테스트에는 코드의 내부 구조의 약간 다른 측면을 테스트하는 데

사용되는 여러 가지 유형이 있음.

* 주요 유형은 다음과 같다.
  1. **경로 테스트 (Path Testing)**:

프로그램의 제어 구조를 기반으로 다양한 경로를 테스트.

* 1. **루프 테스트 (Loop Testing)**:

프로그램 내의 루프가 유효한지 검증.

* 1. **조건 테스트 (Conditional Testing)**:

코드 내의 논리적 조건이 참인지 거짓인지 확인

* 1. **단위 테스트 (Unit Testing)**:

개별 구성 요소와 모듈을 검증.

**화이트 박스 테스트 기법**

* 화이트 박스 테스트를 수행하는 데 사용할 수 있는 다양한 기법.
* 주요 기법.
  1. **문장 커버리지 (Statement Coverage)**:

코드의 실행된 문장 수를 측정.

* 1. **분기 커버리지 (Branch Coverage)**:

각 분기가 최소한 한 번 처리되었는지 확인

* 1. **경로 커버리지 (Path Coverage)**:

프로그램 내의 모든 경로가 최소한 한 번 탐색되었는지 확인.

* 1. **결정 커버리지 (Decision Coverage)**:

모든 결정의 참과 거짓 결과를 검증.

**화이트 박스 테스트의 생애 주기**

* 화이트 박스 테스트는 소프트웨어 개발 생애 주기에서 중요한 단계.
* 테스트의 생애 주기.
  1. **단위 테스트 및 통합 테스트**:

개발 단계에서 수행.

* 1. **기능 테스트**:

시스템 테스트 및 수용 테스트 전에 수행.

* 1. **버그 수정**:

주요 버그를 식별하고 수정하는 기회를 제공.

**수동 및 자동 화이트 박스 테스트**

* 화이트 박스 테스트는 수동 또는 자동으로 수행할 수 있음.
* **수동 화이트 박스 테스트**의 장점:
  1. **깊이 있는 탐색**:

코드의 모든 세부 사항을 탐색할 수 있음.

* 1. **버그 위치 파악**:

버그가 발생한 정확한 위치를 쉽게 찾을 수 있음.

* **자동화 화이트 박스 테스트**의 장점:
  1. **정확성**:

컴퓨터 기반 테스트는 오류의 위험을 줄임.

* 1. **시간 절약**:

자동화된 테스트는 수동 테스트보다 빠름.

* 1. **확장성**:

대규모 테스트를 수행 가능.

**화이트 박스 테스트를 시작하기 위한 요구 사항**

* 화이트 박스 테스트를 시작하기 전에 필요한 모든 것을 준비.
* 주요 요구 사항:
  1. **소스 코드 이해**:

내부 구조에 대한 완전한 이해가 필요.

* 1. **테스트 케이스 작성**:

테스트를 수행하기 위한 개별 지침 세트를 작성.

* 1. **화이트 박스 테스트 도구**:

테스트 자동화를 지원하는 도구를 선택.

**화이트 박스 테스트 프로세스**

* 화이트 박스 테스트는 블랙 박스 테스트보다 시스템의 작동 방식에 대한 더 많은

지식을 요구.

* 주요 프로세스 단계.
  1. **테스트할 기능 식별**:

검증할 기능이나 구성 요소를 결정.

* 1. **모든 가능한 경로 플로우 그래프 작성**:

테스트할 모든 경로를 시각적 표시.

* 1. **테스트 케이스 작성**:

각 경로를 검증하기 위한 테스트 케이스를 작성.

* 1. **테스트 케이스 실행**:

각 테스트 케이스를 실행하고 결과를 기록.

**화이트 박스 테스트의 모범 사례**

* 화이트 박스 테스트의 모범 사례는 수행하는 테스트 유형과 테스트 프로세스의 단계
* 주요 모범 사례:
  1. **테스트 범위 극대화**:

가능한 많은 경로와 분기를 테스트.

* 1. **행동 및 성능 검증**:

시스템의 기능과 성능을 검증하는 테스트 케이스를 작성.

* 1. **독립적인 테스트 케이스 작성**:

각 테스트 케이스가 독립적으로 검증.

**화이트 박스 테스트의 일반적인 실수**

* 화이트 박스 테스트를 시작할 때 개발자가 자주 저지르는 일반적인 실수에 대해

인식하는 것이 중요하다.

* 주요 실수:
  1. **화이트 박스 테스트가 필요 없다고 생각하기**:

블랙 박스 테스트만으로 충분하다고 오해할 수 있음.

* 1. **모든 화이트 박스 테스트를 수동으로 수행하기**:

수동 테스트는 시간이 많이 소요.

* 1. **테스터에게 테스트 케이스 할당하기**:

화이트 박스 테스트는 개발자가 수행.

* 1. **테스트를 서두르기**:

충분한 시간을 할당하지 않으면 테스트 품질이 저하될 수 있음.

**화이트 박스 테스트의 출력 유형**

* 화이트 박스 테스트를 수행할 때 다양한 출력이 생성.
* 주요 출력 유형:
  1. **테스트 결과**:

각 테스트 케이스의 성공 여부를 나타냄.

* 1. **결함**:

발견된 결함이나 버그를 기록.

* 1. **테스트 보고서**:

테스트 결과와 다음 단계에 대한 권장 사항을 포함.

**화이트 박스 테스트의 예시**

* 화이트 박스 테스트는 개발자가 소프트웨어 시스템의 내부 구조가 제대로

작동하는지 확인하는 데 도움을 줌.

* 주요 예시:
  1. **전자상거래 등록 페이지**:

등록 기능이 올바르게 작동하는지 검증.

* 1. **계산기 예시**:

계산기의 내부 계산 과정을 검증하여 정확성을 확인.

**화이트 박스 테스트에서 발견되는 오류**

* 화이트 박스 테스트 중에 시스템 작동에 영향을 미칠 수 있는 버그를 식별하고

위치를 파악할 수 있음.

* 주요 오류 유형:
  1. **논리적 오류**:

프로그램이 논리적으로 작동하지 않는 경우 발생.

* 1. **설계 오류**:

소프트웨어의 논리 흐름과 실제 구현 간의 차이로 인해 발생.

* 1. **타이포그래픽 오류**:

인간의 실수로 인해 발생하는 문법적 오류.

**화이트 박스 테스트의 일반적인 메트릭**

* 화이트 박스 테스트를 수행할 때 일반적인 테스트 메트릭은 테스트의 성공과

포괄성을 측정하는 데 도움이 됨.

* 주요 메트릭:
  1. **코드 커버리지**:

테스트된 코드의 비율을 측정한다.

* 1. **결함 메트릭**:

발견된 결함의 수와 품질을 나타낸다.

* 1. **테스트 실행 메트릭**:

실행된 테스트의 비율을 나타낸다.

* 1. **테스트 지속 시간**:

자동화된 테스트를 실행하는 데 걸리는 시간을 측정한다.

**화이트 박스 테스트 도구**

* 도구와 기술은 화이트 박스 테스트를 훨씬 더 정확하고 효율적이며 포괄적으로

만들 수 있음.

* 주요 도구:
  1. **ZAPTEST**:

화이트 박스 및 블랙 박스 테스트를 자동화할 수 있는 도구이다.

* 1. **Bugzilla**:

결함 추적 및 관리 도구이다.

* 1. **OpenGrok**:

코드베이스를 탐색할 수 있는 오픈 소스 도구이다.

* 1. **SQLmap**:

SQL 인젝션 버그를 탐지하는 도구이다.

**무료 화이트 박스 테스트 도구**

* 비용을 들이지 않고 온라인에서 사용할 수 있는 무료 화이트 박스 테스트 도구가

많이 있다.

* 주요 무료 도구:
  1. **ZAPTEST FREE**:

여러 가상 사용자와 반복을 지원하는 무료 버전.

* 1. **Bugzilla**:

결함 추적 및 관리에 유용한 오픈 소스 도구.

* 1. **OpenGrok**:

코드베이스를 탐색할 수 있는 무료 도구.

**기업 화이트 박스 테스트 도구**

* 보다 큰 기능이나 더 나은 지원을 제공하는 도구가 필요하다면 기업 화이트 박스 테스트 도구가 더 적합할 수 있음.
* 주요 기업 도구:
  1. **ZAPTEST ENTERPRISE**:

자동화 및 RPA 기술을 위한 포괄적인 도구이다.

* 1. **Fiddler**:

웹 애플리케이션을 위한 화이트 박스 테스트 도구이다.

* 1. **HP Fortify**:

보안 취약점을 자동으로 스캔하는 도구이다.

**화이트 박스 테스트 체크리스트**

* 화이트 박스 테스트를 수행할 준비가 되었을 때 시작하기 전에 기억해야 할 사항

목록이 있음.

* 주요 체크리스트 항목:
  1. **자동화 도구 사용**:

테스트 속도를 높이고 오류율을 줄인다.

* 1. **100% 테스트 커버리지 목표**:

가능한 많은 경로와 분기를 테스트해야 한다.

* 1. **명확한 테스트 보고서 작성**:

테스트 결과를 명확하게 기록해야 한다.

**화이트 박스 테스트 결론**

* 소프트웨어 엔지니어링에서 화이트 박스 테스트는 소프트웨어 애플리케이션의

내부 구조와 논리를 검증하는 필수적인 유형의 소프트웨어 테스트.

* 화이트 박스 테스트는 블랙 박스 테스트와 함께 소프트웨어의 기능과

내부 코드의 품질을 확인하는 데 중요한 역할.