정보처리기사 필기

소프트웨어 개발 8 애플리케이션 테스트 관리②

양문자 선생님

출처 : ncs 학습모듈(NCS능력단위 애플리케이션테스트관리)

차례

애플리케이션 통합 테스트

결함관리 도구

- 테스트 결함 관리란 각 단계별 테스트 수행 후 발생한 결함의 재발 방지를 위해, 유사결함 발견 시 시간 단축을 위해 결함을 추적하고 관리하는 활동이다.

결함 관리 프로세스

결함 추이 분석

1. 결함 추이 분석
   (1) 개요
   테스트 완료 후 발견된 결함의 결함 관리 측정 지표의 속성 값들을 분석하고, 향후 애플리케이션의 어떤 모듈 또는 컴포넌트에서 결함이 발생할지를 추정하는 작업
   (2) 분석 유형 : 결함의 분포 분석, 결함 추세 분석, 결함 에이징 분석

2. 결함 관리 측정 지표
   (1) 결함 분포
   각 애플리케이션 모듈 또는 컴포넌트의 특정 속성에 해당하는 결함의 수를 측정하여 결함의 분포를 분석
   (2) 결함 추세
   테스트 진행 시간의 흐름에 따른 결함의 수를 측정하여 결함 추세를 분석
   (3) 결함 에이징
   등록된 결함에 대해 특정한 결함 상태의 지속 시간을 측정하여 분석

애플리케이션 개선 조치사항 작성

테스트 커버리지(Test Coverage)

- 주어진 테스트 케이스에 의해 수행되는 소프트웨어의 테스트 범위를 측정하는 테스트 품질 측정 기준
- 테스트의 정확성과 신뢰성을 향상시키는 역할

1. 기능 기반 커버리지
   - 테스트 대상 애플리케이션의 전체 기능을 모수로 설정하고, 실제 테스트가 수행된 기능의 수를 측정
   - 100% 달성을 목표로 하며, 일반적으로 UI가 많은 시스템의 경우 화면 수를 모수로 사용할 수도 있다.

2. 라인 커버리지(Line Coverage)
   - 애플리케이션 전체 소스 코드의 Line 수를 모수로 테스트 시나리오가 수행한 소스 코드의 Line 수를 측정하는 방법
   - 단위 테스트에서는 이 라인 커버리지를 척도로 삼기도 한다.

3. 코드 커버리지(Code Coverage)
   - 소프트웨어 테스트 충분성 지표 중 하나로 소스 코드의 구문, 조건, 결정 등의 구조 코드 자체가 얼마나 테스트되었는지를 측정하는 방법

   (1) 구문(Statement) 커버리지

   (2) 조건(Condition) 커버리지

   (3) 결정(Decision) 커버리지

   (4) 변형 조건/결정(Modified Condition/Decision) 커버리지
   조건과 결정을 복합적으로 고려한 측정 방법
   결정 포인트 내의 다른 개별적인 조건식 결과에 상관없이 독립적으로 전체 조건식의 결과에 영향을 주는 테스트 커버리지

테스트 결함 식별 및 관리

1. 결함의 식별

(1) 단계별 결함 유입 분류

(가) 기획 시 유입되는 결함
- 사용자 요구사항의 표준 미준수로 인해 테스트 불가능, 요구사항 불명확/불완전/불일치, 기타 결함이 발생

(나) 설계 시 유입되는 결함
- 기획 단계에 유입된 결함 또는 설계 표준 미준수로 인해 테스트 불가능, 기능 설계 불명확/불완전/불일치, 기타 결함이 발생

(다) 코딩 시 유입되는 결함
- 설계 단계에 유입된 결함 또는 코딩 표준 미준수로 인해 기능의 불일치/불완전, 데이터 결함, 인터페이스 결함, 기타 결함이 발생

(라) 테스트 부족으로 유입되는 결함
- 테스트 수행 시 테스트 완료 기준의 미준수, 테스트 팀과 개발 팀의 의사소통 부족, 개발자의 코딩 실수로 인한 결함

테스트 결함 식별 및 관리

(2) 결함 심각도별 분류

: 애플리케이션에 발생한 결함이 어떤 영향을 끼치며, 그 결함이 얼마나 치명적인지를 나타내는 척도

(가) 결함 심각도 분류 사례

- 치명적(Critical) 결함, 주요(Major) 결함, 보통(Normal) 결함, 경미한(Minor) 결함, 단순(Simple) 결함

(나) 결함 심각도 관리

- 결함 관리의 정확성과 신뢰성 향상을 위해 결함 심각도의 각 단계별로 표준화된 용어를 사용하여 정의하여야 하며
- 프로젝트 및 조직 차원에서 결함 관리 활동을 수행해야 한다.

테스트 결함 식별 및 관리

(3) 결함 우선순위

(가) 결함 심각도와 결함 우선순위

- 결함 우선순위는 발생한 결함이 얼마나 빠르게 처리되어야 하는지를 결정하는 척도
- 결함 심각도가 높아도 우선순위가 반드시 높은 것은 아니며, 애플리케이션의 특성에 따라 우선순위가 결정 될 수 있다.

(나) 결함 우선순위 표현 사례

- 결정적(Critical), 높음(High), 보통(Medium), 낮음(Low) 또는 즉시 해결, 주의 요망, 대기, 개선 권고 등으로 분류할 수 있다.

2. 결함 관리 항목

: 테스트 수행 후 발견된 결함은 결함 관리 시스템에 등록하여 관리해야 하며, 등록 시 다음 항목들은 필수로 등록한다.

(1) 결함 내용
(2) 결함 ID
(3) 결함 유형
(4) 발견일
(5) 심각도
(6) 우선순위(결함 수정의 우선순위)
(7) 시정 조치 예정일
(8) 수정 담당자
(9) 재테스트 결과
(10) 종료일

테스트 자동화 도구

: 테스트 도구를 활용하여 반복적인 테스트 작업을 스크립트 형태로 구현함으로써, 테스트 시간 단축과 인력 투입 비용을 최소화하는 한편, 쉽고 효율적인 테스트를 수행할 수 있는 방법이다.

테스트 도구의 장점

(가) 반복되는 테스트 데이터 재입력 작업의 자동화
(나) 사용자 요구 기능의 일관성 검증에 유리
(다) 테스트 결과 값에 대한 객관적인 평가 기준 제공
(라) 테스트 결과의 통계 작업과 그래프 등 다양한 표시 형태 제공
(마) UI가 없는 서비스의 경우에도 정밀한 테스트 가능

테스트 도구의 단점

(가) 도구 도입 후 도구 사용 방법에 대한 교육 및 학습이 필요
(나) 도구를 프로세스 단계별로 적용하기 위한 시간, 비용, 노력이 필요
(다) 상용 도구의 경우 고가, 유지 관리 비용이 높아 추가 투자가 필요

테스트 자동화 도구 유형

1. 정적 분석 도구(Static Analysis Tools)
   (가) 만들어진 애플리케이션을 실행하지 않고 분석하는 방법
   (나) 소스 코드에 대한 코딩 표준, 코딩 스타일, 코드 복잡도 및 남은 결함을 발견하기 위하여 사용
   (다) 테스트를 수행하는 사람이 작성된 소스 코드에 대한 이해를 바탕으로 도구를 이용해서 분석하는 것

2. 테스트 실행 도구(Test Execution Tools)
   - 테스트를 위해 작성된 스크립트를 실행한다.
   - 작성된 스크립트는 각 스크립트마다 특정 데이터와 테스트 수행 방법을 포함하고 있다.

   (가) 데이터 주도 접근 방식
      - 테스트 데이터를 스프레드시트에 저장하고, 이 데이터를 읽고 실행할 수 있도록 한다.

   (나) 키워드 주도 접근 방식
      - 테스트를 수행할 동작을 나타내는 키워드와 테스트 데이터를 스프레드시트에 저장한다.

3. 성능 테스트 도구(Performance Test Tools)

- 애플리케이션의 처리량, 응답 시간, 경과 시간, 자원 사용률에 대해 가상의 사용자를 생성하고 테스트를 수행함으로써 성능 목표를 달성하였는지를 확인하는 도구

4. 테스트 통제 도구(Test Control Tools)

- 테스트 통제 도구에는 테스트 계획 및 관리를 위한 **테스트 관리 도구**, 테스트 수행에 필요한 데이터와 도구를 관리하는 **형상 관리 도구**, 테스트에서 발생한 결함에 대해 관리 하거나 협업을 지원하기 위한 **결함 추적/관리 도구** 등이 있다.
- 조직의 요구사항에 최적화된 형태의 정보를 생성, 관리하기 위하여 스프레드시트 등 **다른 도구들과 연계**하여 사용할 수도 있다.

5. 테스트 장치(Test Harness)

(가) 정의
애플리케이션 컴포넌트 및 모듈을 테스트하는 환경의 일부분으로, 테스트를 지원하기 위한 코드와 데이터를 말하며, 단위 또는 모듈 테스트에 사용하기 위해 코드 개발자가 작성한다.

(나) 구성요소

1) 테스트 드라이버(Test Driver)
- 테스트 대상 하위 모듈을 호출하고, 파라미터를 전달하고, 모듈 테스트 수행 후의 결과를 도출하는 등 **상향식** 테스트에 필요하다.

2) 테스트 스텁(Test Stub)
- 제어 모듈이 호출하는 타 모듈의 기능을 단순히 수행하는 도구로 **하향식** 테스트에 필요하다.

3) 테스트 슈트(Test Suites)
- 테스트 대상 컴포넌트나 모듈, 시스템에 사용되는 테스트 케이스의 집합을 말한다.

4) 테스트 케이스(Test Case)
- 입력 값, 실행 조건, 기대 결과 등의 집합을 말한다.

5) 테스트 스크립트(Test Script)
- 자동화된 테스트 실행 절차에 대한 명세를 말한다.

6) 목 오브젝트(Mock Object)
- 사용자의 행위를 조건부로 사전에 입력해 두면, 그 상황에 예정된 행위를 수행하는 객체를 말한다.

소프트웨어 개발

테스트 단계별 테스트자동화 도구

테스트 단계	자동화 도구	도구 설명
테스트 계획	요구사항 관리	• 고객 요구사항 정의 및 요구사항 관리를 지원
테스트 분석/설계	테스트케이스 생성	• 테스트 기법에 따른 테스트 케이스 작성과 테스트 데이터 생성을 지원
테스트 수행	테스트 자동화	• 기능 테스트와 UI 테스트 등 단위 테스트 및 통합테스트를 지원
	정적 분석	• 코딩표준, 런타임 오류 등을 검증
	동적 분석	• 대상시스템 시뮬레이션을 통한 오류 검출
	성능 테스트	• 부하생성기 등을 이용하여 가상 사용자를 생성하고, 시스템의 처리능력을 측정하는 도구
	모니터링	• 시스템 자원(CPU, Memory 등)의 상태 확인 및 분석 지원

테스트 단계별 테스트자동화 도구

테스트 단계	자동화 도구	도구 설명
테스트 관리	커버리지 측정	• 테스트 완료 후 테스트 충분성 여부 검증 지원
	형상 관리	• 테스트 수행에 필요한 도구, 데이터 및 문서 관리
	결함 추적/관리	• 테스트에서 발생한 결함 추적 및 관리 활동 지원

통합 테스트

1. 개요

(1) 통합 테스트의 개념

- 애플리케이션 통합 테스트는 소프트웨어 각 모듈 간의 인터페이스 관련 오류 및 결함을 찾아내기 위한 체계적인 테스트 기법이다.

(2) 통합 테스트의 목적

- 단위 테스트가 끝난 모듈 또는 컴포넌트 단위의 프로그램이 설계 단계에서 제시한 애플리케이션과 동일한 구조와 기능으로 구현된 것인지를 확인하는 것이다.

(3) 통합 테스트 수행 방법의 분류

- 비점증적인 방식 : 모든 컴포넌트를 사전에 통합하여 전체 프로그램을 한꺼번에 테스트하는 것
- 점증적인 방법 : 상향식 통합, 하향식 통합 방식

2. 하향식 통합(Top Down)

(1) 방식

메인 제어 모듈(프로그램)로부터 아래 방향으로 제어의 경로를 따라 이동하면서 하향식으로 통합하면서 테스트를 진행하며, 메인 제어 모듈에 통합되는 하위 모듈과 최하위 모듈은 '깊이-우선' 또는 '너비-우선' 방식으로 통합된다.

(2) 수행 단계

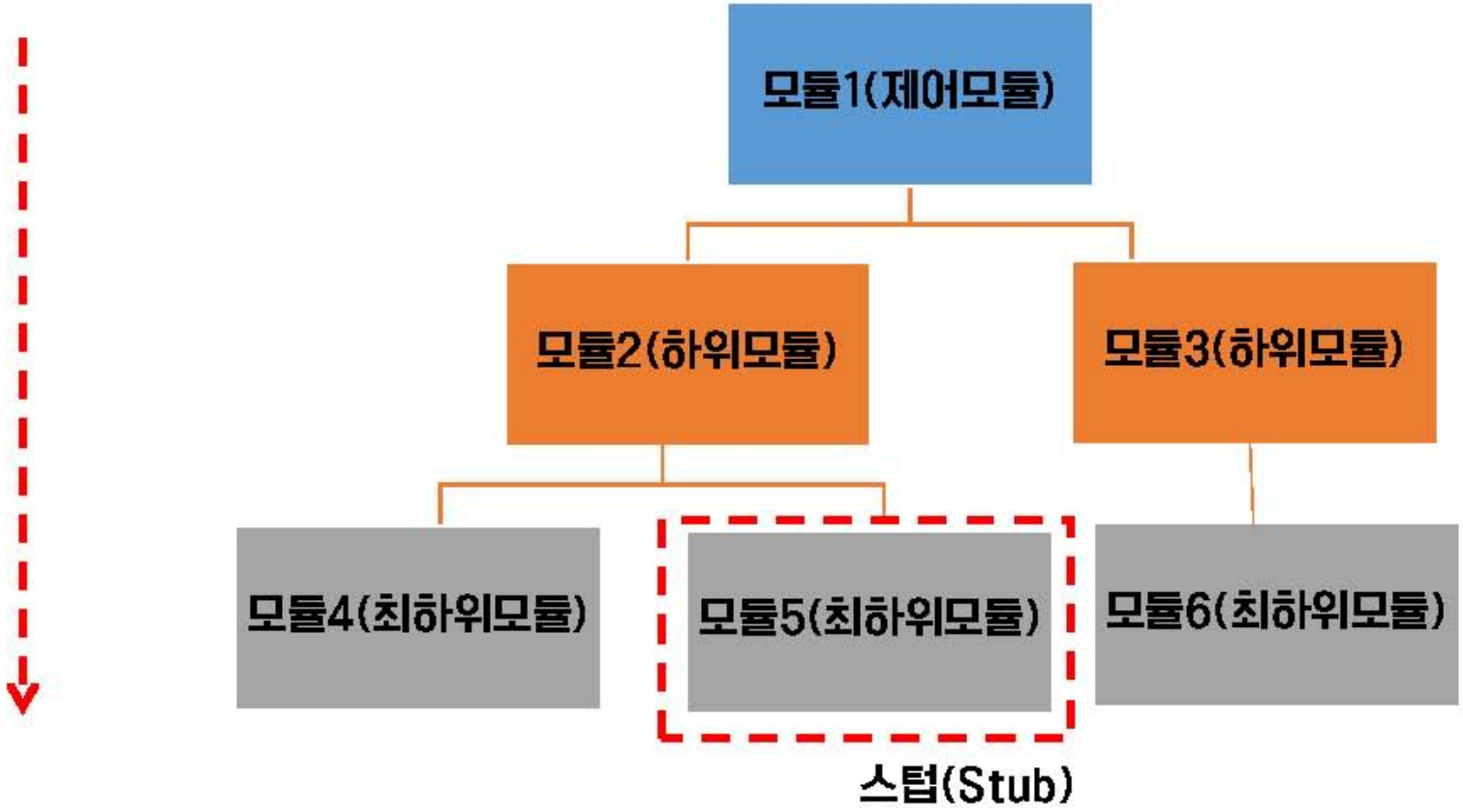
(가) 메인 제어 모듈은 작성된 프로그램을 사용하고, 아직 작성되지 않은 하위 제어 모듈 및 모든 하위 컴포넌트를 대신하여 더미 모듈인 스텁(Stub)을 개발한다.

(나) 깊이-우선 방식 또는 너비-우선 방식에 따라, 하위 모듈인 스텁이 한 번에 하나씩 실제 모듈로 대체된다.

(다) 각 모듈 또는 컴포넌트를 통합하면서 테스트가 수행된다.

(라) 테스트가 완료되면 스텁이 실제 모듈 또는 컴포넌트로 작성된다.

하향식 방식의 통합 테스트



3. 상향식 통합(Bottom Up)

(1) 방식

애플리케이션 구조에서 최하위 레벨의 모듈 또는 컴포넌트로부터 위쪽 방향으로 제어의 경로를 따라 이동하면서 구축과 테스트를 시작한다.

(2) 수행 단계

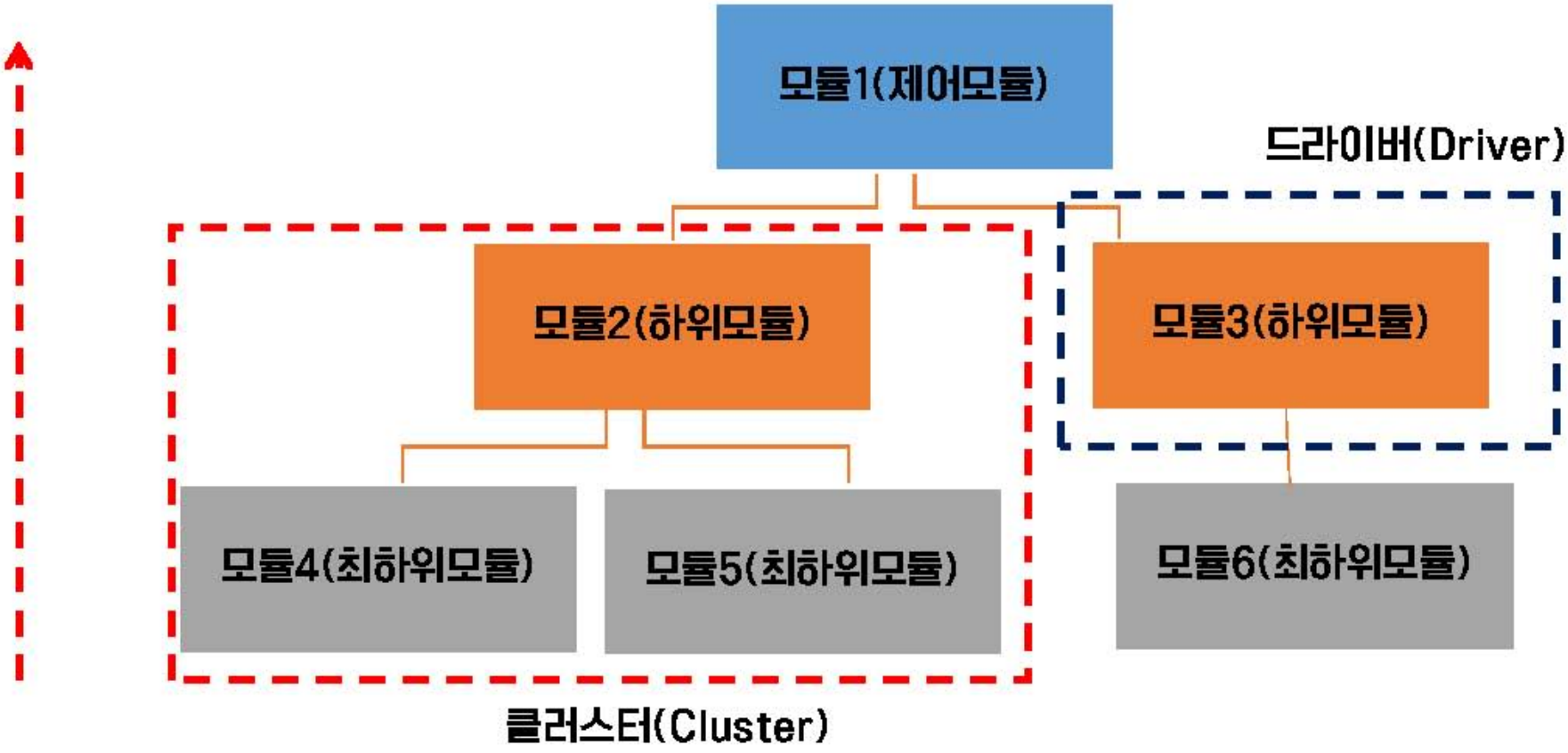
(가) 최하위 레벨의 모듈 또는 컴포넌트들이 하위 모듈의 기능을 수행하는 클러스터(Cluster)로 결합

(나) 상위의 모듈에서 데이터의 입력과 출력을 확인하기 위한 더미 모듈인 드라이버(Driver)를 작성

(다) 각 통합된 클러스터 단위를 테스트한다.

(라) 테스트가 완료되면 각 클러스터들은 프로그램의 위쪽으로 결합되며, 드라이버는 실제 모듈 또는 컴포넌트로 대체된다.

상향식 방식의 통합 테스트



회귀 테스팅(Regression Testing)

통합 테스트가 완료된 후에 변경된 모듈이나 컴포넌트가 있다면 새로운 오류 여부를 확인하기 위해 회귀 테스트를 수행할 수 있다.

(1) 정의

(가) 통합 테스트 과정에서 오류를 제거하거나 수정한 프로그램이 새로운 형태의 오동작이나 오류를 일으킬 수 있다.

(나) 회귀 테스트는 모듈이나 컴포넌트의 변화로 인해 의도하지 않은 오류가 생기지 않았음을 보증하기 위해 반복 테스트하는 것을 말한다.

(2) 회귀 테스트 케이스 선정 방법

(가) 모든 애플리케이션의 기능을 수행할 테스트 케이스의 대표적인 샘플을 도출한다.

(나) 변경에 의한 영향도가 가장 높은 애플리케이션 기능에 집중한 추가적인 테스트 케이스를 도출한다.

(다) 실제 수정이 발생한 모듈 또는 컴포넌트에서부터 시행하는 테스트 케이스를 도출한다.

문제풀이

- 하향식 통합에 있어서 모듈 간의 통합 시험을 위해 일시적으로 필요한 조건만을 가지고 임시로 제공되는 시험용 모듈을 무엇이라고 하는가?

① Stub
② Driver
③ Procedure
④ Function

(2020년 1, 2회 정보처리기사 필기 기출문제 소프트웨어 개발)

문제풀이

- 다음이 설명하는 애플리케이션 통합 테스트 유형은?

> - 깊이 우선 방식 또는 너비 우선 방식이 있다.
> - 상위 컴포넌트를 테스트 하고 점증적으로 하위 컴포넌트를 테스트 한다.
> - 하위 컴포넌트 개발이 완료되지 않은 경우 스텁(Stub)을 사용하기도 한다.

① 하향식 통합 테스트
② 상향식 통합 테스트
③ 회귀 테스트
④ 빅뱅 테스트

(2020년 3회 정보처리기사 필기 기출문제 소프트웨어 개발)

애플리케이션 성능 개선

소스 코드 품질 분석 도구의 이해

- 소스 코드에 대한 코딩 스타일, 설정된 코딩 표준, 코드의 복잡도, 코드 내에 존재하는 메모리 누수 현황, 스레드의 결합 등을 발견하기 위하여 사용하는 분석 도구

1. 정적 분석 도구
   작성된 소스 코드를 실행시키지 않고, 코드 자체만으로 코딩 표준 준수 여부, 코딩 스타일 적정 여부, 잔존 결함 발견 여부를 확인하는 코드 분석 도구이다.

2. 동적 분석 도구
   애플리케이션을 실행하여 코드에 존재하는 메모리 누수 현황을 발견하고, 발생한 스레드의 결함 등을 분석하기 위한 도구이다.

소프트웨어 개발

소스코드 품질 분석 도구

구분	도구명	도구 설명
정적 분석 도구	pmd	자바 및 타 언어 소스코드에 대한 버그, 데드코드 분석
	cppcheck	C/C++ 코드에 대한 메모리누수, 오버플로우 등 문제 분석
	SonarQube	소스코드 품질 통합플랫폼, 플러그인 확장 가능
	checkstyle	자바 코드에 대한 코딩 표준 준수 검사 도구
코드 복잡도	ccm	다양한 언어의 코드 복잡도 분석 도구, Linux, Mac 환경 CLI 형태 지원
	cobertura	jcoverage 기반의 테스트 커버리지 측정도구
동적 분석 도구	Avalanche	Valgrind 프레임워크 및 STP 기반 소프트웨어 에러 및 취약점 동적 분석 도구
	Valgrind	자동화된 메모리 및 쓰레드 결함 발견 분석 도구

문제풀이

- **소스코드 품질분석 도구 중 정적분석 도구가 아닌 것은?**

① pmd ② checkstyle
③ valance ④ cppcheck

(2020년 4회 정보처리기사 필기 기출문제 소프트웨어 개발)

- **소스코드 품질분석 도구 중 정적분석 도구가 아닌 것은?**

① pmd ② cppcheck
③ valMeter ④ checkstyle

(2020년 1, 2회 정보처리기사 필기 기출문제 소프트웨어 개발)