동국대학교
dongguk university

제출일	2023.05.26	학과	컴퓨터공학전공
과목	S/W품질관리및테스팅	학번	2018112007
담당교수	이호정 교수님	이름	이승현

1. 소프트웨어 품질을 향상하기 위해 고려할만한 내용은 무엇인가?

아래의 내용을 기준으로 설명하시오.(30 pts)

- ISO/IEC 9126, 14598, 12119 또는 이들의 개선 표준인 ISO/IEC 25000시리즈
- ISO/IEC 15504(그와 비슷한 성격의 CMMi)
- 소프트웨어의 품질과 테스팅과의 관련성

가) ISO/IEC 9126, 14598, 12119

a. ISO/IEC 9126

(1) 소프트웨어 품질 특성과 척도에 관한 표준 지침으로, 고객 관점에서 소프트웨어에 관한 품질특 성과 품질 하위 특성을 정의

(2) 계층 구조의 품질 모형을 채택하고 있어, 기능성, 신뢰성, 사용성, 효율성, 유지보수성, 이식성의 6가지 품질 특성을 정의하며, 각각의 품질 특성에 대한 여러 개의 하위 특성을 포함하고 있음.

(3) 각 품질 하위 특성별로 세부 척도를 제시하고 있으며, 소프트웨어 개발과정에서 개발자들이 적용할 수 있는 내부 척도와 소프트웨어 사용자들이 개발 초기 또는 개발 완료 후에 적용할 수 있는 외부 척도로 구성

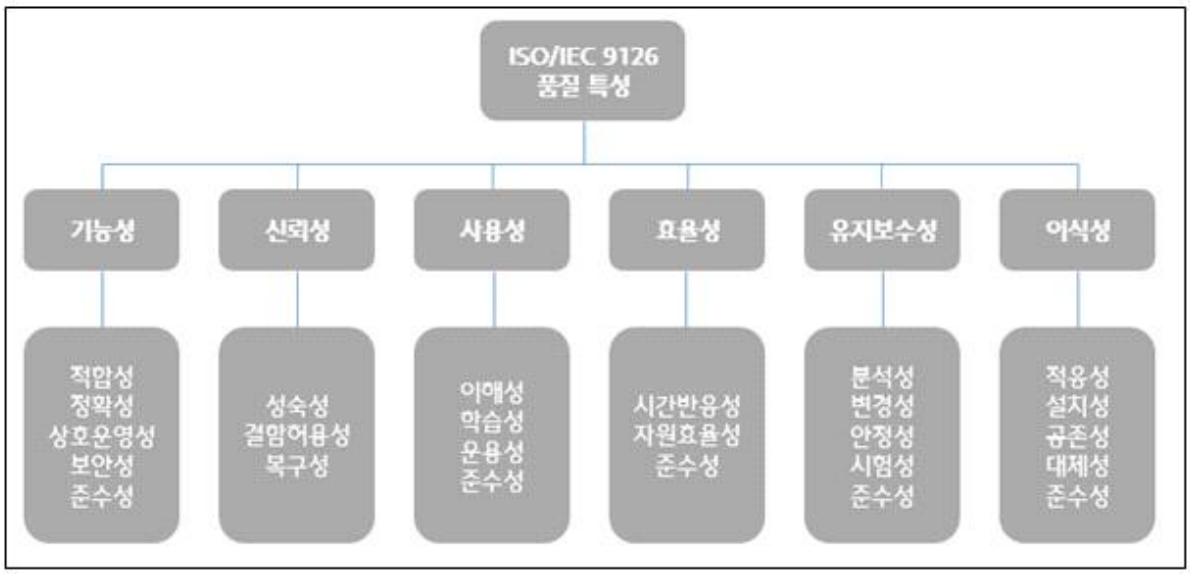
(4) 내부 척도는 설계/코드와 관련된 소프트웨어의 내부 속성을 측정하여 제품에 적용하는 척도

(5) 외부 척도는 소프트웨어가 사용될 때의 외부적 성질을 표현하며, 실행 가능한 소프트웨어의 시험/운영으로 측정

(6) 소프트웨어 제품에 대한 품질 요구사항을 기술하는 데 사용할 수 있으며, 개발 중이거나 개발 완료된 소프트웨어의 품질을 측정하는 척도로 사용할 수 있음

(7) 1991년에 제정된 후, 1994년부터 품질 특성과 내부 척도, 외부 척도를 조정하고, 품질 측정 절차를 ISO/IEC 14598 표준으로 분리됨.

(8) 소프트웨어 품질 특성



b. ISO/IEC 14598

(1) 개발 중이거나 완성된 소프트웨어 제품을 개발자, 사용자, 제삼자가 평가자 관점에서 객관적으로 평가하기 위한 국제 표준

(2) ISO/IEC 9126 기반으로 하는 사용 절차, 기본 상황 및 소프트웨어 평가 프로세스

(3) 특징

구분	설명
반복성	제품에 대한 평가자가 동일 사양으로 평가 시 동일한 것으로 간주할 수 있는 결과가 도출
재생산성	제품에 대한 평가자가 다른 데이터로 측정했을 때 동일한 것으로 간주할 수 있는 결과가 발생
공평성	목적의식을 갖고 평가 결과를 임의로 유도하면 안됨
객관성	평가자의 가정, 사견을 받아들이지 않아야 함

(4) 구성항목

ISO/IEC 14598-1	General Overview (일반 개요)	- 범위와 용어를 정의, 14598 - 9126 관계 설명 - 전체적인 로드맵 제시	전체 포함
ISO/IEC 14598-2	Planning and Management (계획과 관리)	- 품질 측정 계획의 준비와 구현 - 제품 평가 기능을 지원하기 위한 요구사항과 안내 지침을 포함한 전체적인 사항 제공	평가 지원
ISO/IEC 14598-3	Process for Developers (개발자를 위한 프로세스)	- 개발 단계, 개발자 준수 사항 - 개발자가 품질 평가 사용하는 방법을 제공	평가 프로세스
ISO/IEC 14598-4	Process for Acquirers (구매자를 위한 프로세스)	- 제품을 구매하기 위한 계획을 수립할 때 사용 - 구매과정에서 품질 평가할 수 있는 방법 제공	평가 프로세스
ISO/IEC 14598-5	Process for Evaluators (평가자를 위한 프로세스)	- 품질 전문가를 위한 프로세스 - 평가자가 개발 과정, 최종 SW 품질 평가 사용	평가 프로세스
ISO/IEC 14598-6	Documentation of Evaluation Modules (평가 모듈을 위한 문서)	- 개발자, 구매자, 평가자가 품질을 평가할 때 평가 모델에 대한 기본적인 가이드와 이론적인 모델 제공 - 문서화하고 검증하는 지침 제공	평가 지원

c. ISO/IEC 12119

(1) 정보 기술, 소프트웨어 패키지 제품에 대한 일반적인 품질 요구사항 시험을 위한 국제 표준

(2) 규격 내용은 품질, 지침, 세부 인증 등이고, 요건 사항은 명확화, 유사문서 정의, 변경성, 환경명세, 보안성, 기타 등등

(3) 소프트웨어 패키지는 제품 설명서, 사용자 문서, 프로그램으로 구성

(4) 테스트 대상은 제품 명세서, 사용자 메뉴얼, 사용자 요구 테스트 결과서 등이 포함되며 프로그램 코드가 필요한 구조적 테스트는 제외

(5) 품질 요구사항

평가 대상	평가 내용	항목 수
제품설명서	- 내용에 관한 일반적인 요구사항 - 식별과 지시 - 기능성에 대한 설명 - 신뢰성에 대한 설명 - 사용성에 대한 설명 - 효율성에 대한 설명 - 유지보수성에 대한 설명 - 이식성에 대한 설명	22
사용자 문서	완전성, 정확성, 일관성, 이해성, 개괄성	10
SW 제품 및 데이터	제품 품질 속성	12

(6) 평가 절차

단계	내용
제품설명서 시험	제품설명서에 대한 요구사항, 권고사항의 수행에 대한 시험
사용자 문서 시험	사용자 문서에 대한 요구사항, 권고사항의 수행에 대한 시험
실행 프로그램 시험	프로그램, 데이터에 관한 요구사항, 권고사항에 대한 시험
시험 기록	시험 반복하기, 충분한 정보를 포함한 기록 작성
시험 보고서 작성	시험의 목적과 결과 요약

나) ISO/IEC 15504

(1) SPICE(Software Process Improvement Capability Determination)라고도 함

(2) 소프트웨어 프로세스 전반을 망라한 심사를 실시하여 조직의 소프트웨어 개발 프로세스를 개선하고 개발자의 개발 능력을 향상시킴으로써 개발 위험을 통제하기 위한 목적으로 ISO에서 추진하는 소프트웨어 품질 표준화 심사 평가 모형으로 소프트웨어 프로세스 전반을 망라하여 심사를 하고 그 결과에 따른 조직의 프로세스를 개선하여 나가는 활동에 대한 표준화 방법

(3) ISO 12207 소프트웨어 생명주기의 프로세스를 포함하는 프로세스와 프로세스 능력을 2차원으로 평가하는 모델

(4) 소프트웨어 사업자의 능력 평가 수단으로 사용 가능

(5) 다수의 프로세스 심사 모델인 CMM, ISO 9000등의 장점을 수용한 통합 모델

(6) SPICE 평가 방식

1) ISO/IEC 15504는 CMM과 마찬가지로 조직의 프로세스를 개선하기 위한 활동을 지원하기 위하여 현재의 프로세스 상태를 파악하여 성숙한 능력 수준을 측정

2) SPICE에서 정의하고 있는 프로세스는 5개의 카테고리로 구분되며 세부 프로세스는 40개로 정의. 또한 이들의 능력 수준은 수준별 측정관점에 따라 6개의 수준으로 구분.

<카테고리>

i. CUS(고객-공급자 프로세스 범주) : 소프트웨어를 개발하여 고객에게 전달하는 것을 지원하고, 소프트웨어를 정확하게 운용하고 사용하도록 하기 위한 프로세스로 구성

ii. ENG(공학 프로세스 범주) : 시스템과 소프트웨어 제품을 직접 명세화, 구현, 유지 보수하는 프로세스로 구성

iii. SUP(지원 프로세스 범주) : 소프트웨어 생명주기에서 다른 프로세스(지원 프로세스 포함)에 의해 이용되는 프로세스로 구성

iv. MAN(관리 프로세스 범주) : 소프트웨어 생명 주기에서 프로젝트 관리자에 의해 사용되는 프로세스로 구성

v. ORG(조직 프로세스 범주) : 조직의 업무 목적을 수립하고, 조직이 업무 목표를 달성하는 데 도움을 주는 프로세스로 구성되어 있다.

<수준>

단계	측정관점
Level 0(불완전 수준)	프로세스가 구현되지 않거나 프로세스 목적을 달성하지 못함
Level 1(수행 수준)	해당 프로세스의 목적은 달성하지만, 계획되거나 추적되지 않음
Level 2(관리 수준)	프로세스 수행이 계획되고 관리되어 작업 산출물이 규정된 표준과 요구에 부합
Level 3(확립 수준)	표준 프로세스를 사용하여 계획되고 관리
Level 4(예측 가능 수준)	표준 프로세스 능력에 대하여 정량적인 이해와 성능이 예측
Level 5(최적 수준)	정의된 프로세스와 표준 프로세스가 지속해서 개선

다) 소프트웨어의 품질과 테스팅과의 관련성

소프트웨어의 품질과 테스팅은 밀접한 관련이 있다. 품질은 소프트웨어가 사용자의 요구를 충족시키는 정도를 나타내며, 테스팅은 소프트웨어의 품질을 평가하고 개선하기 위해 수행되는 활동이다. 소프트웨어의 품질과 테스팅 간의 관련성은 다음과 같다.

1) 품질 특성 도출: 소프트웨어 품질은 사용자의 요구를 만족시키는 다양한 특성으로 설명될 수 있다. 이러한 품질 특성은 기능성, 신뢰성, 사용성, 효율성, 유지보수성, 이식성 등으로 구성될 수 있다. 테스팅은 이러한 품질 특성을 기반으로 소프트웨어의 동작을 확인하고, 품질 목표에 대한 충족 여부를 평가한다.

2) 결함 탐지: 테스팅은 소프트웨어에서 결함(버그)을 발견하고 해결하기 위해 수행된다. 결함은 소프트웨어의 품질을 저하시키는 요소로 작용할 수 있다. 테스팅은 결함을 식별하고 수정함으로써 소프트웨어의 품질을 향상시킬 수 있다.

3) 품질 관리: 테스팅은 소프트웨어의 품질 관리에 핵심적인 역할을 수행한다. 테스트 계획, 테스트 케이스 설계, 실행 및 결과 분석은 품질을 지속적으로 모니터링하고 개선하기 위해 필요한 활동이다. 테스팅을 통해 얻은 결과는 소프트웨어 개발자와 관리자에게 품질에 대한 정보를 제공하고, 필요한 조치를 취할 수 있도록 돕는다.

4) 사용자 만족도: 테스팅은 사용자의 요구사항을 충족시키는 소프트웨어를 개발하기 위한 중요한 수단이다. 만족스러운 사용자 경험을 제공하기 위해서는 품질이 높은 소프트웨어가 필요하다. 테스팅을 통해 결함을 최소화하고 사용자 요구사항을 충족하는 소프트웨어를 개발할 수 있다.

따라서 소프트웨어의 품질과 테스팅은 서로 긴밀한 관련성을 가지고 있다. 테스팅은 소프트웨어 품질을 평가하고 개선하기 위한 필수적인 활동으로, 품질 관리와 사용자 만족도 향상을 위해 반드시 고려되어야 한다.

2. Validation과 Verification이 어떤 것인지 간략히 기술하시오.

- Verification: 소프트웨어가 설계된 요구사항과 일치하는지 여부를 확인하는 과정이다. 주로 개발 초기 단계에서 수행된다. Verification은 소프트웨어가 정확하게 구현되었는지, 코드가 요구사항과 일치하는지, 설계 문서와 일치하는지 등을 확인하는 작업이다. 이는 소프트웨어가 정확하게 작동하고 기능이 올바르게 구현되었는지를 확인하는 과정이다.

- Validation: 소프트웨어가 사용자의 요구사항을 충족시키는지를 확인하는 과정이다. 주로 개발 완료 후에 수행된다. Validation은 실제로 사용자가 소프트웨어를 사용하며 기대한 결과를 얻을 수 있는지 확인하는 작업이다. 이는 소프트웨어가 사용자의 요구사항을 정확히 충족시키고 목표에 부합하는지를 검증하는 과정이다

3. White-box Testing과 Black-box Testing은 무엇인가?

- White-box Testing: 소프트웨어의 내부 구조와 동작을 이해하고 테스트하는 기법이다. 개발자 관점에서 테스트를 수행하는 경우에 주로 사용된다. White-box Testing은 코드의 흐름, 제어 구조, 조건문, 루프 등을 분석하고, 코드의 모든 경로를 테스트하여 올바른 동작과 예외 상황을 확인한다. 이는 소프트웨어의 내부 동작을 살펴보고 구조적인 결함을 발견하는 데 초점을 둔다. 주요 기법으로는 제어 구문 검사, 루프 검사, 데이터 흐름 검사, 기본 경로 검사가 있다.

- Black-box Testing: 소프트웨어의 내부 동작을 몰라도 기능적인 측면을 테스트하는 기법이다. 사용자나 외부 테스터 관점에서 테스트를 수행하는 경우에 주로 사용된다. Black-box Testing은 소프트웨어의 입력과 출력을 검증하여 예상된 결과를 확인한다. 테스터는 소프트웨어의 내부 구조를 알 필요 없이 사양, 요구사항, 사용자 인터페이스 등을 기반으로 테스트 케이스를 설계하고 실행한다. 이는 소프트웨어의 기능적인 부분을 중심으로 테스트하는 데 초점을 둔다. 주요 기법으로는 동등 분할, 경계 값 분석, 원인-효과 그래프, 상태 전이 테스트 등이 있다.

4. 정적 분석, 리뷰, 인스펙션, 워크쓰루가 무엇인지 각각을 비교하여 기술하시오.

- 정적 분석: 소스 코드나 문서를 대상으로 구문적, 의미적, 규칙적인 오류를 검출하는 기법이다. 프로그램을 실행하지 않고 코드 자체를 분석하여 잠재적인 결함을 식별하고 예방한다. 주로 자동화된 도구를 사용하여 수행된다. 정적 분석은 코드의 복잡성, 일관성, 코딩 규칙 준수 여부 등을 평가하며, 시간적으로 효율적이고 반복적으로 적용할 수 있다. 하지만 코드 실행 경로에 대한 정보는 제한적이므로 동적인 오류는 검출하기 어렵다.

- 리뷰: 팀 구성원들이 모여 소프트웨어의 문서, 코드, 설계 등을 통찰력을 바탕으로 검토하고 피드백을 주고받는 활동이다. 리뷰는 주로 수동적으로 수행되며, 구성원들이 서로의 작업을 검사하여 결함을 발견하고 품질을 향상시킨다. 리뷰는 통찰력과 경험에 의존하여 설계 결함, 오류, 일관성 부족 등을 식별한다. 리뷰는 개발 초기부터 종료 후까지 다양한 문서와 코드를 검토할 수 있다.

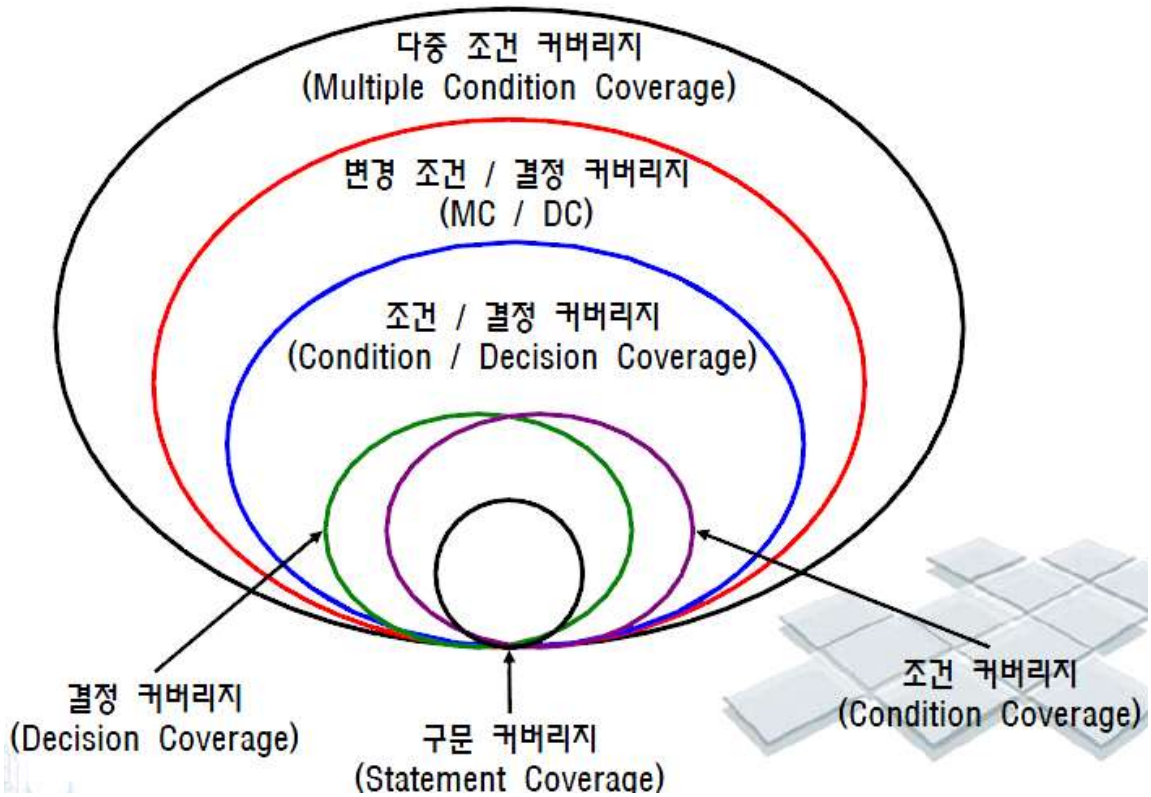
- 인스펙션: 리뷰의 한 유형으로, 정형화된 프로세스를 따라 수행되는 검토 기법이다. 리더와 구성원들이 사전에 정의된 역할과 지침에 따라 소프트웨어 문서나 코드를 검토한다. 인스펙션은 검토의 효율성과 품질 향상을 위해 명확한 절차를 따르며, 결함 식별 및 수정을 위한 체계적인 접근 방법을 제공한다. 일관성, 오류, 누락된 요구사항 등을 검증한다.

- 워크쓰루: 소프트웨어의 문서나 코드를 검토하는 과정에서 설명과 토론을 중심으로 진행되는 활동이다. 팀 구성원들이 모여 소프트웨어를 차례대로 훑어가면서 개발 과정을 설명하고 피드백을 주고받는다. 워크쓰루는 개발 초기 단계에서 빠르게 진행되며, 이해도를 높이고 설계 결함이나 개선점을 발견하는 데에 도움을 준다. 워크쓰루는 비구조적이고 유연한 방식으로 진행되며, 구성원들 간의 의사소통과 학습 기회를 제공한다.

5. 커버리지의 종류와 커버리지 사이의 포함관계를 그림으로 설명하시오.

종류	설명
구문 커버리지 (Statement)	- 프로그램을 구성하는 모든 문장이 최소한 한 번은 실행될 수 있는 입력 데이터를 테스트 데이터로 선정하는 기준 - 최종적으로 실행된 구문이 몇 퍼센트인지 측정하는 것으로 다른 커버리지에 비해 가장 약하다. - 프로그램 상에 존재하는 가능한 경우를 모두 검증하지 못한다. $문장\ 커버리지(\%) = \frac{테스트\ 케이스\ 집합에\ 의해\ 실행된\ 문장의\ 수}{전체\ 실행\ 가능한\ 프로그램\ 문장의\ 수} \times 100$
결정 커버리지 (Decision)	- Branch Coverage라고도 함. - 결정 포인트 내의 전체 조건식이 최소한 참(True) 한번, 거짓(False)이 한 번씩 선택되었는지 특정하는 기준 - 개별 조건식의 개수와 상관없이 테스트 케이스의 최소 개수는 2개 - 조건/결정 커버리지에 비해 약함 $결정\ 커버리지(\%) = \frac{테스트\ 케이스\ 집합에\ 의해\ 실행된\ 결정문의\ 결과\ 수}{전체\ 프로그램의\ 결정문의\ 결과\ 수} \times 100$
조건 커버리지 (Condition)	- 전체 조건식의 결과와 관계없이 개별 조건식이 참 한번, 거짓 한번을 모두 갖도록 개별 조건식을 조합한 형태의 커버리지 - 조건 커버리지와 결정 커버리지는 서로 포용하지 않는다. $조건\ 커버리지(\%) = \frac{테스트\ 케이스\ 집합에\ 의해\ 실행된\ 개별\ 조건의\ 결과\ 수}{전체\ 프로그램의\ 개별\ 조건의\ 결과\ 수} \times 100$
조건/결정 커버리지 (Condition/Decision)	전체 조건식의 결과가 참 한번, 거짓 한번을 갖도록 각 개별조건식을 조합하는데, 이때 각 개별 조건식도 참과 거짓을 한 번씩 모두 갖도록 조합하는 것으로 결정 커버리

	지와 조건 커버리지를 포함하는 커버리지 $결정/조건\ 커버리지(\%) = \frac{테스트\ 케이스\ 집합에\ 의해\ 실행된\ 결정문과\ 개별\ 조건의\ 결과\ 수}{전체\ 프로그램의\ 결정문과\ 개별\ 조건의\ 결과\ 수} \times 100$
변경조건/결정 커버리지 (MC/DC)	- 개별 조건식이 다른 개별 조건식에 무관하게 전체 조건식의 결과에 독립적으로 영향을 주도록 함으로써 조건/결정 커버리지를 향상한 것으로 결정, 조건/결정 커버리지보다 강력 $MCDC\ 커버리지(\%) = \frac{테스트\ 케이스\ 집합에\ 의해\ MCDC를\ 만족하는\ 조건의\ 개수}{총\ 조건의\ 개수} \times 100$
다중조건 커버리지 (Multiple Condition)	- 결정 포인트 내에 있는 모든 개별 조건식의 모든 가능한 논리적인 조합을 고려하여 100% 커버리지를 보장 $다중\ 조건\ 커버리지(\%) = \frac{테스트\ 케이스\ 집합에\ 의해\ 실행된\ 조건의\ 조합\ 수}{전체\ 프로그램의\ 개별\ 조건의\ 조합\ 수} \times 100$



- 결정 커버리지는 구문 커버리지를 포함.
- 조건 커버리지는 구문 커버리지를 포함.
- 결정 커버리지와 조건 커버리지는 서로 포용하지 못함.
- 조건 / 결정 커버리지는 결정 커버리지와 조건 커버리지, 그 이하 요소를 포함.
- 변경 조건 / 결정 커버리지는 조건 / 결정 커버리지, 그 이하 요소를 포함.
- 다중 조건 커버리지는 변경 조건 / 결정 커버리지, 그 이하 요소를 포함.

6. Integration test에서 backbone, Big-Bang, Bottom-up, Top-down의 특징과 장단점을 기술하시오.

(1) Backbone
<특징>
- 시스템의 중요한 기능이나 핵심 모듈을 우선적으로 통합하여 테스트한다. 나머지 모듈은 임시적인 스텁 또는 드라이버를 사용하여 대체한다.

<장점>
- 핵심 기능이 우선 테스트 되므로 중요한 결함을 조기에 발견할 수 있다.
- 개발 초기 단계부터 시스템의 핵심 기능을 확인할 수 있어 일부 기능의 동작이 확인되어 사용자나 이해관계자들의 신뢰도를 높일 수 있다.

<단점>
- 나머지 모듈이 통합되지 않은 상태로 테스트 되므로, 전체 시스템 동작에 대한 테스트가 미흡할 수 있다.
- 스텁과 드라이버의 사용으로 인해 초기 테스트를 위한 추가 작업과 관리가 필요하다.

(2) Big-Bang
<특징>
- 개별 모듈들을 독립적으로 테스트한 후, 모든 모듈을 한꺼번에 통합하여 전체 시스템을 테스트한다.

<장점>
- 개발 초기 단계에 모듈들을 독립적으로 개발하고 테스트할 수 있다.
- 모든 모듈을 한꺼번에 통합하여 테스트하므로, 전체 시스템 동작을 실제로 확인할 수 있다.

<단점>
- 모든 모듈을 통합하기 전까지는 시스템의 동작을 확인할 수 없으므로 결함을 발견하기까지의 시간이 오래 걸릴 수 있다.
- 모든 모듈이 동시에 통합되므로, 결함의 원인을 파악하고 디버깅하기가 어려울 수 있다.

(3) Bottom-up
<특징>
- 모듈의 하위 수준 모듈부터 시작하여 상위 수준 모듈로 순차적으로 통합하여 전체 시스템을 테스트한다.

<장점>
- 하위 수준 모듈부터 테스트하므로, 의존성이 큰 모듈들이 먼저 통합되어 결함을 조기에 발견할 수 있다.
- 하위 수준 모듈들의 테스트 결과가 확인되므로, 개발 진척 상황을 실시간으로 확인할 수 있다.

<단점>
- 상위 수준 모듈이 아직 통합되지 않은 상태에서 하위 수준 모듈을 테스트하므로, 전체 시스템 동작에 대한 테스트가 미흡할 수 있다.
- 스텁과 드라이버의 사용으로 인해 추가 작업과 관리가 필요하다.

(4) Top-down
<특징>
- 상위 수준 모듈부터 시작하여 하위 수준 모듈로 순차적으로 통합하여 전체 시스템을 테스트한다.

<장점>
- 상위 수준 모듈부터 테스트하므로, 전체 시스템 동작을 빠르게 확인할 수 있다.
- 상위 수준 모듈들의 테스트 결과가 확인되므로, 기능의 동작이 보장된 후 하위 모듈들을 통합하여 테스트할 수 있다.

<단점>
- 하위 수준 모듈이 아직 통합되지 않은 상태에서 상위 수준 모듈을 테스트하므로, 의존성이 있는 하위 모듈의 테스트가 지연될 수 있다.
- 스텁과 드라이버의 사용으로 인해 추가 작업과 관리가 필요하다.

7. 결정포인트 DPoint = A && B || C 에 대한 질문이다.

(1) 다중 커버리지(MCC)를 만족하는 테스트 케이스를 테이블의 형태로 작성한 다음, MC/DC (변형 조건/결정 커버리지)를 만족하는 테스트 케이스를 선별하시오.
(2) MC/DC를 만족하는 테스트 케이스가 된 이유를 해설에 기술하시오.
※ MC/DC를 만족하는 테스트 케이스들이 두 개 이상 존재할 수 있다는 사실을 기반으로 하여 서술하면 쉽다. 서술이 완전하지 않으면 0점으로 채점한다.

다중 커버리지(MCC) 테스트 케이스 테이블:

테스트 케이스	DPoint	A	B	C	해설
1	True	True	True	True	MC/DC를 만족하지 않는다.
2	True	True	True	False	MC/DC를 만족한다. A를 False로 변경하면 전체 조건식의 값이 False로 변한다. 조건식 A에 대해 테스트 케이스 6과 페어를 이룬다. B를 False로 변경하면 전체 조건식의 값이 False로 변한다. 조건식 B에 대해 테스트 케이스 4와 페어를 이룬다.
3	True	True	False	True	MC/DC를 만족한다. C를 False로 변경하면 전체 조건식의 값이 False로 변한다. 조건식 C에 대해 테스트 케이스 4와 페어를 이룬다. MC/DC를 만족하는 최소의 테스트 케이스 개수는 4개로 MC/DC를 만족할 수 있는 조건 묶음을 찾아보면 A:{2, 6}, B: {2, 4}, C: {3, 4}이므로 {2, 3, 4, 6}이라는 MC/DC를 만족하는 최소 개수의 테스트 케이스 조합을 찾을 수 있다. **일반적으로 N개 입력을 가진 결정에서 N+1개의 테스트 케이스 필요하다.**
4	False	True	False	False	MC/DC를 만족한다. B를 True로 변경하면 전체 조건식의 값이 True로 변한다. 조건식 B에 대해 테스트 케이스 2와 페어를 이룬다. C를 True로 변경하면 전체 조건식의 값이 True로 변한다. 조건식 C에 대해 테스트 케이스 3과 페어를 이룬다.
5	True	False	True	True	MC/DC를 만족한다. C를 False로 변경하면 전체 조건식의 값이 False로 변한다. 조건식 C에 대해 테스트 케이스 6과 페어를 이룬다. MC/DC를 만족하는 최소의 테스트 케이스 개수는 4개로 MC/DC를 만족할 수 있는 조건 묶음을 찾아보면 A:{2, 6}, B: {2, 4}, C: {5, 6}이므로 {2, 4, 5, 6}이라는 MC/DC를 만족하는 최소 개수의 테스트 케이스 조합을 찾을 수 있다. **일반적으로 N개 입력을 가진 결정에서 N+1개의 테스트 케이스 필요**
6	False	False	True	False	MC/DC를 만족한다. A를 True로 변경하면 전체 조건식의 값이 True로 변한다. 조건식 A에 대해 테스트 케이스 2와 페어를 이룬다. C를 True로 변경하면 전체 조건식의 값이 True로 변한다.

					조건식 C에 대해 테스트 케이스 5와 페어를 이룬다.
7	True	False	False	True	MC/DC를 만족한다. C를 False로 변경하면 전체 조건식의 값이 False로 변한다. 조건식 C에 대해 테스트 케이스 7과 페어를 이룬다. MC/DC를 만족하나 최소 개수의 테스트 케이스에는 포함되지 않는다. 왜냐하면 조건 A, B에 대해 다른 테스트 케이스와 겹치지 않기 때문이다.
8	False	False	False	False	MC/DC를 만족한다. C를 True로 변경하면 전체 조건식의 값이 True로 변한다. 조건식 C에 대해 테스트 케이스 8과 페어를 이룬다. MC/DC를 만족하나 최소 개수의 테스트 케이스에는 포함되지 않는다. 왜냐하면 조건 A, B에 대해 다른 테스트 케이스와 겹치지 않기 때문이다.